VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MOBILNÍ APLIKACE PRO 3D REKONSTRUKCI
MOBILE APPLICATION FOR 3D RECONSTRUCTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Krátký**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Miloslav Richter, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Martin Krátký ***ID:*** 195672

Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Mobilní aplikace pro 3D rekonstrukci

POKYNY PRO VYPRACOVÁNÍ:

Realizujte aplikaci pro 3D rekonstrukci scény na mobilní platformě.

1. Nastudujte problematiku mobilních aplikací a zpracování obrazu v souvislosti s 3D rekonstrukcí.
2. Stanovte parametry měřených scén pro výsledné řešení. Pořiďte testovací sady pro testování agloritmů.
3. Popište vhodné platformy a zvolte jednu pro kterou vytvoříte aplikaci.
4. Navrhněte a realizujte algoritmy pro zpracování pořízených dat.
5. Realizujte mobilní aplikaci využívající navržené algoritmy.
6. Proveďte zhodnocení kvality navržených algoritmů.

DOPORUČENÁ LITERATURA:

Gonzalez R.C., Woods R.E.: Digital Image Processing, 4th edition, Pearson, 2017, ISBN 978-0133356724

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 17.5.2021

Vedoucí práce: Ing. Miloslav Richter, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem diplomové práce je vytvoření mobilní aplikace sloužící k prostorové rekonstrukci. V práci jsou nejprve popsány principy zpracování obrazu vhodné pro řešení tohoto problému. Dále jsou uvedeny dostupné platformy pro vytvoření mobilní aplikace a popsány parametry měřených scén. Pro platformu Android je vytvořena mobilní aplikace, která je následně otestována rekonstrukcí objektů v rozdílných podmínkách.

KLÍČOVÁ SLOVA

Android, mobilní aplikace, prostorová rekonstrukce, stereoskopie, struktura z pohybu, zpracování obrazu

ABSTRACT

The aim of this master thesis is to create mobile application for spatial reconstruction. Thesis describes image processing methods suitable for solving this problem. Available platforms for creating mobile application are described. The parameters of the measured scenes are defined. A mobile application containing the described methods is created. The application is tested by reconstruction of objects in different conditions.

KEYWORDS

Android, mobile application, spatial reconstruction, stereoscopy, structure from motion, image processing

KRÁTKÝ, Martin. *Mobilní aplikace pro 3D rekonstrukci*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2021, 70 s. Diplomová práce. Vedoucí práce: Ing. Miloslav Richter, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Martin Krátký
VUT ID autora:	195672
Typ práce:	Diplomová práce
Akademický rok:	2020/21
Téma závěrečné práce:	Mobilní aplikace pro 3D rekonstrukci

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Miloslavu Richterovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Mobilní zařízení mají v dnešní době dostatečný výkon, aby na nich mohly běžet algoritmy zpracování obrazu potřebné k prostorové rekonstrukci. V bakalářské práci [1] byl vytvořen program pro vytvoření prostorového modelu pomocí snímků zaznamenávajících nasvícený laserový vzor na snímaném objektu. Výsledné modely dosahují vysoké přesnosti, nicméně je nutné podstoupit proceduru pořizování snímků při nasvícení laserovým vzorem. Pořizování a zpracování dat tímto způsobem je časově náročné. Cílem této práce je vytvoření aplikace, pomocí které bude možné uskutečnit 3D rekonstrukci ze snímků na mobilním zařízení. Jedná se tak o praktické a méně časově náročné řešení.

Tato práce je rozdělena do kapitol a každá z nich popisuje potřebné části k vytvoření mobilní aplikace pro prostorovou rekonstrukci. První polovina práce je teoretická a druhá polovina popisuje praktické využití teoretických poznatků. První kapitola popisuje projektivní geometrii. Jsou v ní popsány parametry kamery. Kamera slouží k pořizování dat snímané scény. Dále je uvedena epipolární geometrie, která popisuje vzájemný vztah mezi dvěma pozicemi kamer a scénou. Následující kapitola se věnuje metodám prostorové rekonstrukce. Je zde popsána metoda struktura z pohybu, pomocí které je ze snímků zaznamenaných z více pohledů vytvořeno mračno bodů. V další části kapitoly je popsána metoda více-pohledová stereoskopie, která slouží k zahuštění tohoto mračna pro zvýšení počtu získaných 3D bodů. V závěru kapitoly je popsána polygonální rekonstrukce povrchu, při které je z mračna bodů vytvořen polygonální povrch reprezentující výsledný 3D model. Následující kapitola je věnována mobilním platformám, které jsou vhodné pro realizaci mobilní aplikace. V této kapitole je popsán význam realizace aplikace zpracování obrazu na mobilních platformách. Ve zbytku kapitoly jsou popsány platformy Android a iOS. Poslední kapitola teoretické části popisuje proces získávání potřebných dat. V úvodní části kapitoly jsou popsány parametry scény, které mají vliv na kvalitu rekonstrukce a je popsán vhodný způsob pro pořizování dat. Ve zbylé části kapitoly jsou uvedeny snímače mobilních zařízení, pomocí kterých lze získat dodatečné informace, které jsou užitečné při prostorové rekonstrukci. V úvodní kapitole praktické části je popsán algoritmus prostorové rekonstrukce využívající metody uvedené v teoretické části práce. V následující kapitole je popsána realizace mobilní aplikace. Je zde uvedena struktura aplikace a jsou popsány obrazovky, ze kterých je aplikace složena. V závěru kapitoly je popsáno, jakým způsobem byly připojeny potřebné závislosti a jaká jsou cílená zařízení. Poslední kapitola práce se zabývá testováním vytvořené aplikace. Aplikace je otestována rekonstrukcí dvou objektů v rozdílných podmínkách. V průběhu kapitoly je zhodnocena kvalita získaných modelů.

1 Projektivní geometrie

Cílem prostorové rekonstrukce je vytvořit 3D model scény, která je zaznamenána na snímcích z více pohledů kamer. Při prostorové rekonstrukci jsou využity poznatky o vzájemném vztahu více pohledů se scénou a vlastnosti kamer. Matematickým aparátem geometrie více pohledů je projektivní geometrie [2]. Snímač, pomocí kterého jsou zaznamenány informace o scéně je kamera. Pořízené snímky kamerou jsou dvourozměrné a při jejich pořízení se ztrácí hloubková informace o zaznamenané 3D scéně. Cílem epipolární geometrie je definovat vztah mezi snímky dvou pohledů zachycujících stejnou scénu a body v prostoru. Pomocí epipolární geometrie je tak možné získat souřadnice bodů v prostoru ze snímků zachycujících scénu a vlastností kamer.

V následující kapitole budou popsány principy používané při prostorové rekonstrukci pomocí snímků získaných z více pohledů pomocí kamery. Nejprve budou uvedeny parametry kamery definující její vlastnosti a vztah s okolní scénou. Následně bude popsána epipolární geometrie, která definuje vztah mezi dvěma snímky a scénou, kterou zachycují. Dále bude uveden princip triangulace, který se používá při výpočtu bodů v prostoru. V závěru kapitoly bude uvedeno speciální rozložení pohledů nazývané paralelní rozložení.

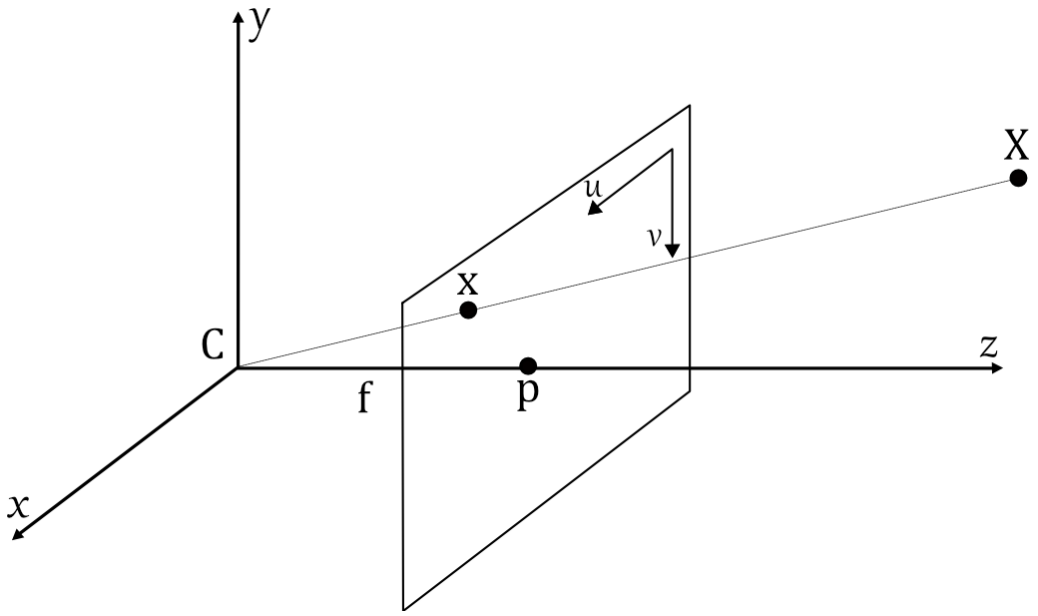
1.1 Parametry kamery

Kamera je důležitým nástrojem při prostorové rekonstrukci. Zachycuje snímanou scénu na snímky složené z pixelů, které jsou následně zpracovány pro vytvoření prostorového modelu. Vlastnosti kamery definují vztah mezi daty zaznamenanými na snímku a reálnou scénou, kterou snímek zachycuje. Používané parametry jsou model kamery a vnitřní a vnější parametry kamery. Tyto informace jsou obvykle získány pomocí procesu zvaného kalibrace kamery. V kapitole 4 budou uvedeny alternativní způsoby získání parametrů kamery.

1.1.1 Model kamery

Pro definování vlastností kamery se používá model dírkové kamery. Model dírkové kamery uvažuje kameru s tenkou čočkou. Tato aproximace je vhodná pro většinu aplikací zpracování obrazu [2]. Kamera provádí transformaci z 3D prostoru na dvourozměrný snímek. Tomuto jevu se také říká perspektivní projekce [3]. Vlastnosti kamery je možné popsat pomocí vnitřních a vnějších parametrů kamery. Na Obr. 1.1 je znázorněn model kamery. C je pozice kamery v prostoru, f je ohnisková vzdálenost definující vzdálenost zobrazovací roviny snímku od středu promítání. X je bod v

prostoru, který je zobrazený na snímku jako x. Souřadný systém na snímku se značí jako (u, v) a p je střed snímku.



Obr. 1.1: Model kamery [1]

1.1.2 Vnitřní parametry kamery

Vnitřní parametry kamery definují vlastnosti charakteristické pro danou kameru. Tyto parametry se nemění při změně pozice a orientace kamery. Vnitřní parametry kamery mohou být definovány výrobcem, případně je možné je získat pomocí kalibrace kamery. Kalibrace je vhodná i v případě, kdy jsou parametry definovány výrobcem pro získání aktuálních hodnot. Vnitřní parametry kamery lze vyjádřit formou matice, která má následující tvar:

$$K = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.1}$$

kde (u_0, v_0) je střed snímku. Měřítko v ose u udává f_u a f_v udává měřítko v ose v [2]. Tyto hodnoty jsou si obvykle rovny a odpovídají ohniskové vzdálenosti f. Alternativní variantou vnitřních parametrů kamery jsou normalizované vnitřní parametry, ve kterých je uvažována ohnisková vzdálenost rovna jedné. Tento tvar je používán v případě, kdy nejsou k dispozici kalibrační parametry. Tvar matice je následující:

$$K_n = \begin{bmatrix} 1 & 0 & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.2}$$

1.1.3 Vnější parametry kamery

Vnější parametry kamery udávají pozici a orientaci kamery. Vnější parametry je možné vyjádřit pomocí rotace a translace. Translační vektor T vyjadřuje změnu pozice kamery. Je složen ze tří hodnot, které udávají změnu pozice v souřadnicích (x, y, z). Translační vektor má následující tvar:

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{1.3}$$

kde (t_x, t_y, t_z) udává posun po osách (x, y, z). Změna hodnot translačního vektoru má vliv na pozici kamery v prostoru a na souřadný systém snímku [2]. Rotační matice R vyjadřuje rotace kolem souřadných os (x, y, z). Její tvar je následující:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{1.4}$$

kde r_{ij} vyjadřuje prvek rotační matice i-tého sloupce na j-tém řádku [4]. Pro názornost významu prvků rotační matice jsou v následujících rovnicích uvedeny rotace kolem souřadných os (x, y, z):

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{bmatrix}, \tag{1.5}$$

$$R_y(\alpha) = \begin{bmatrix} cos(\alpha) & 0 & -sin(\alpha) \\ 0 & 1 & 0 \\ sin(\alpha) & 0 & cos(\alpha) \end{bmatrix}, \tag{1.6}$$

$$R_z(\alpha) = \begin{bmatrix} cos(\alpha) & sin(\alpha) & 0 \\ -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1.7}$$

kde R_x odpovídá rotaci kolem osy x, R_y kolem osy y a R_z kolem osy z [5]. V případě kombinování více rotací je výsledná matice závislá na pořadí v jakém byly rotace uskutečněny.

1.1.4 Matice kamery

V předchozích sekcích byly uvedeny vnitřní a vnější parametry kamery. Kombinací těchto parametrů vznikne matice kamery. Matice kamery je v literatuře také nazývána jako projekční matice. Projekční matice P je definována následujícím vztahem:

$$P = K\,[R \mid -RT]. \tag{1.8}$$

Oddělovač | znamená, že výsledná matice je složena ze dvou submatic [2].

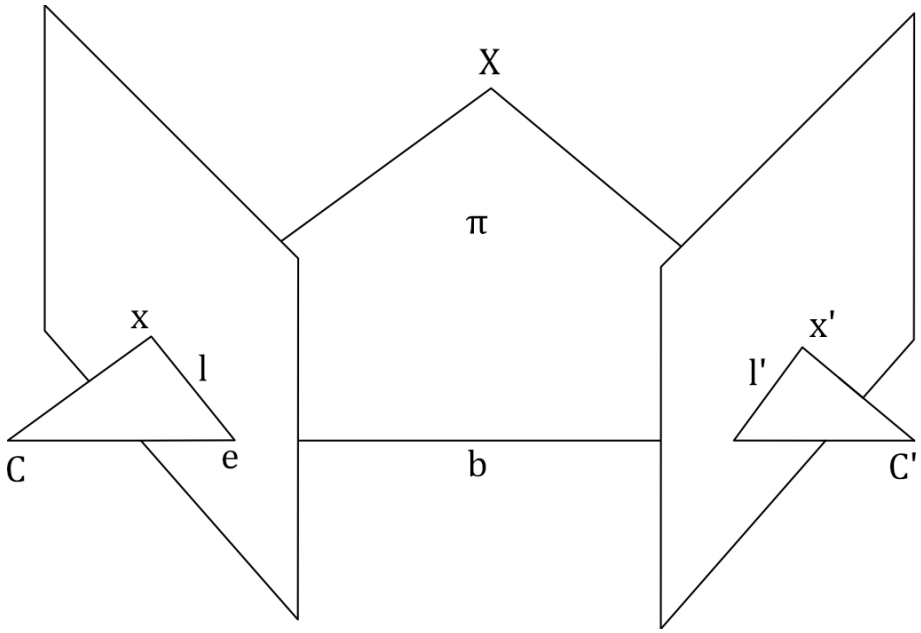
Využití matice kamery je pro transformaci bodů z prostoru na snímek. Tento jev se nazývá projekce. Obdobně transformace ze souřadného systému snímku do prostoru se nazývá zpětná projekce. Projekce z prostoru na snímek je docíleno použitím následujícího vztahu:

$$x = PX, \tag{1.9}$$

kde x jsou souřadnice bodu na snímku a X je bod v prostoru [2].

1.2 Epipolární geometrie

Geometrie dvou pohledů zaznamenávajících stejnou scénu se nazývá epipolární geometrie. Epipolární geometrie vyjadřuje vzájemnou závislost dvou pohledů a jejich vztah s body v prostoru. Matematicky je epipolární geometrie definována pomocí fundamentální a esenciální matice. Z pravidel epipolární geometrie vychází metoda prostorové rekonstrukce nazývána triangulace.



Obr. 1.2: Epipolární geometrie [1]

Na Obr. 1.2 je grafické znázornění epipolární geometrie. Bod v prostoru X je průsečíkem dvou optických paprsků. Tyto paprsky prochází centrem kamery C a pozicí bodu na snímku x. Centra kamer a bod v prostoru tvoří plochu, která je

nazývána epipolární plocha π. Jejím průsečíkem se snímky pohledů jsou definovány epipolární přímky l. Epipolární přímky prochází bodem x daného pohledu a epipolém e. Epipól je bod, ve kterém se protnou veškeré epipolární přímky daného pohledu. Při paralelním rozložení kamer (kap. 1.2.3) jsou epipolární přímky obou pohledů souhlasně orientované a protnou se v nekonečnu. Vzdálenost mezi centry kamer je báze b. Délka báze má vliv na přesnost a počet viditelných bodů pohledy. Zvýšení délky báze zvýší přesnost, nicméně se sníží počet viditelných bodů oběma pohledy.

1.2.1 Fundamentální a esenciální matice

Fundamentální a esenciální matice jsou algebraickou reprezentací epipolární geometrie [6]. Jejich hlavní využití je pro hledání korespondujících bodů a pro získání vnějších parametrů kamer. K tomu slouží rovnice vyjadřující vztah mezi korespondujícími body a fundamentální maticí. Rovnice se často nazývá epipolární omezení a má následující tvar:

$$x'^T F x = 0, \tag{1.10}$$

kde F je fundamentální matice, x je bod na prvním pohledu a x' je jemu odpovídající bod na druhém pohledu [6]. Pokud jsou známy vnitřní parametry kamery, je možné získat esenciální matici. Esenciální matice je vyjádřena v normalizovaných souřadnicích na rozdíl od fundamentální matice, která je vyjádřena v pixelech. Esenciální matici je možné určit z fundamentální matice a vnitřních parametrů kamer pomocí následujícího vztahu:

$$E = K'^T F K, \tag{1.11}$$

kde E je esenciální matice, F je fundamentální matice, K jsou vnitřní parametry prvního pohledu a K' jsou vnitřní parametry druhého pohledu [6].

1.2.2 Triangulace

V předchozí sekci byl uveden princip epipolární geometrie. Pravidel epipolární geometrie využívá princip používaný při prostorové rekonstrukci nazývaný triangulace. Centra kamer tvoří s bodem v prostoru viditelným z obou pohledů trojúhelník. Při znalosti souřadnic těchto bodů na snímku, báze a úhlů, které svírají pohledy lze pomocí pravidel trigonometrie určit souřadnice bodu v prostoru.

1.2.3 Paralelní rozložení pohledů

Speciální konfigurací dvou pohledů je paralelní rozložení. V tomto případě jsou snímky pohledů navzájem paralelní. Důsledkem tohoto rozložení je, že epipolární

přímky odpovídají řádkům na snímcích. Body na snímcích obou pohledů tak mají společnou souřadnici v. Detekce korespondujících bodů je tak v tomto případě redukována na jednu dimenzi. Rozdíl mezi pozicí bodu na prvním pohledu u a totožným bodem na druhém pohledu u' se nazývá disparita d a je definována následujícím vztahem:

$$d = u - u'. \tag{1.12}$$

Z disparity lze vypočítat 3D souřadnice daného bodu pomocí následujícího vztahu:

$$x = \frac{-b(u+u')}{2d},\ y = \frac{bv}{d},\ z = \frac{bf}{d}, \tag{1.13}$$

kde b je báze, f je ohnisková vzdálenost a v je souřadnice bodu na snících [2]. Body s nízkou disparitou jsou ve velké vzdálenosti od snímků. Přesnost vypočtené hloubky závisí na velikosti disparity a nízká velikost disparity způsobuje vyšší chybu. Z toho důvodu je vhodné zvolit dostatečně velkou bázi a nastavit konfiguraci tak, aby scéna zahrnula co největší část snímku. Pár snímků, který není v paralelním rozložení lze převést na paralelní rozložení pomocí procesu, který se nazývá rektifikace snímků.

2 Prostorová rekonstrukce

V předchozí kapitole byla popsána projektivní geometrie, která definuje pravidla používaná při prostorové rekonstrukci. Tato kapitola se zabývá metodami prostorové rekonstrukce, jejichž principy fungují na základě těchto pravidel. V tomto textu jsou uvažovány pasivní metody prostorové rekonstrukce. Pasivní metody zpracovávají snímky zachycující pouze daný objekt. Při použití aktivních metod je na daný objekt nasvícen vzor pomocí projektoru, nebo laseru. Aktivní metody dosahují přesnějších výsledků, nicméně pro použití v mobilní aplikaci jsou více vhodné pasivní metody.

V úvodu této kapitoly bude popsána metoda struktura z pohybu. Tato metoda slouží k určení bodů v prostoru ze sady snímků zachycujících stejnou scénu. Použitím této metody jsou také získány vnější parametry pohledů a je často používána právě pro zjištění těchto parametrů před dalším zpracováním. Aplikováním struktury z pohybu vznikne řídké mračno bodů. V další části této kapitoly je popsána metoda více-pohledová stereoskopie, která pomocí dat získaných při použití struktury z pohybu zahustí mračno bodů. Závěrem kapitoly je popsán způsob, jakým lze z mračna bodů vytvořit polygonální plochu, která tvoří výsledný 3D model.

2.1 Struktura z pohybu

Struktura z pohybu je metoda prostorové rekonstrukce, při které je scéna zaznamenána z více pozic. Metoda se často používá v případě, kdy nejsou k dispozici vnější parametry kamer. Princip metody struktury z pohybu vychází z epipolární geometrie. Ze všech zaznamenaných snímků se vytvoří dvojice, které představují páry, na které lze aplikovat pravidla epipolární geometrie. Pro každý pohled se detekují významné body na snímku a následně se analyzuje, zda odpovídají bodům z ostatních snímků. Tento proces se nazývá sledování vlastností a slouží k propojení nalezených bodů mezi všemi snímky. Při dostatečném počtu nalezených bodů je možné získat pozice a orientace pohledů. Zpracováním získaných dat pomocí triangulace jsou získány body v prostoru. Při určování zmíněných parametrů dochází k reprojekční chybě. Tato chyba je minimalizována pomocí optimalizačního algoritmu bundle adjustment.

2.1.1 Sledování vlastností

Sledování vlastností je proces, při kterém je hledán vztah mezi zaznamenanými snímky. Na snímcích jsou detekovány významné body. Jedná se o body nesoucí informaci. Významné body by měly být reprodukovatelné, to znamená že by měly být detekovatelné i z jiných pohledů. Významné body představují například hrany, rohy

a ostré změny v textuře. Body jsou detekovány pomocí detektorů. Detektory jsou algoritmy hledající významné body a liší se mezi sebou výpočetní náročností a robustností vůči deformacemi snímků. Zvolený detektor by měl být co nejvíce nezávislý vůči rotaci, translaci, měřítku a intenzitě osvětlení [19]. Vlastnosti významných bodů jsou popsány pomocí příznakových vektorů a jejich porovnáním jsou nalezeny body, které si odpovídají. Těmto bodům se říká korespondující body. Souhrn korespondujících bodů všech pohledů se nazývá sledování vlastností a může být realizován formou struktury definující souřadnice korespondujících bodů na všech snímcích. Na Obr. 2.1 je zobrazen příklad bodů na sousedních snímcích, které mohou být použity pro vytvoření sledování vlastností.

Důležitou částí při realizaci sledování vlastností je detekce bodů. V této fázi rekonstrukce je preferována reprodukovatelnost bodů na úkor jejich počtu. K detekci významných bodů jsou používány algoritmy SIFT [20], SURF [21] a KAZE [22], případně modifikovaná verze pro snížení výpočetní náročnosti AKAZE [23]. Tyto detektory jsou invariantní vůči translaci, rotaci a měřítku. Liší se mezi sebou způsobem detekce bodů a strukturou příznakového vektoru. Příznakový vektor hraje důležitou roli při hledání korespondujících bodů sousedních snímků. Algoritmus SURF byl vytvořen jako alternativa s nižší výpočetní náročností k algoritmu SIFT. Detektory SIFT a SURF jsou charakteristické detekováním vyššího množství bodů, zatímco u detektoru AKAZE je vyšší reprodukovatelnost bodů [24].

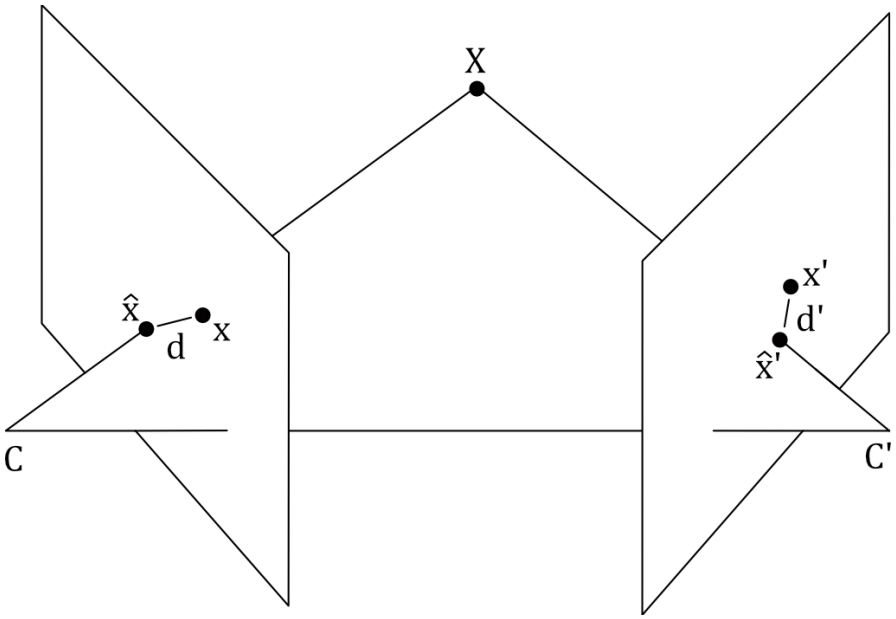
Obr. 2.1: Příklad korespondujících bodů na sousedních snímcích

2.1.2 Určení parametrů kamer a bodů v prostoru

Metoda struktura z pohybu je často používána v případě, kdy jsou k dispozici pouze snímky zachycující scénu a nejsou známy vnější parametry kamer. V tomto případě je nutné nejprve určit tyto parametry a až poté určit body v prostoru. Alternativní způsoby určení vnějších parametrů kamer budou popsány v kapitole 4. Při určování parametrů kamer a bodů v prostoru se zpracovávají data získaná při sledování vlastností. Nejprve se zpracuje pár snímků s nejvyšším počtem korespondujících bodů. Pro tento pár se určí vnější parametry kamer z korespondujících bodů. Pro určení parametrů je možné použít pěti-bodý algoritmus [18]. V případě, že se podaří určit vnější parametry kamer jsou vypočteny body v prostoru pomocí triangulace. V tuto chvíli je již možné aplikovat optimalizační algoritmus bundle adjustment. Následně se zpracuje další pár, přičemž pozice jednoho z pohledů je již známa z předchozího výpočtu [25]. Tento postup se opakuje pro všechny dvojice pohledů.

2.1.3 Bundle adjustment

V předchozích sekcích bylo uvedeno, jakým způsobem je možné určit body v prostoru a matice kamer ze snímků zaznamenaných z více pohledů. Přesnost získaných parametrů závisí na způsobu detekce bodů a způsobu jakým byly určeny matice kamer. Při určování zmíněných parametrů vznikají nepřesnosti, které způsobují reprojekční chybu (Obr. 2.2). Velikosti reprojekční chyby odpovídá vzdálenost mezi pozicí rekonstruovaného bodu na snímku a bodem získaným zpětnou projekcí rekonstruovaného bodu na snímek.



Obr. 2.2: Reprojekční chyba (vychází z [6])

Bundle adjustment [7] je optimalizační algoritmus sloužící k minimalizaci reprojekční chyby. Cílem tohoto algoritmu je získat parametry, pro které je vztah 1.9 platný a minimalizovat vzdálenost mezi naměřeným bodem v prostoru a bodem získaným zpětnou projekcí pro každý snímek, na kterém je bod viditelný [6]. Jedná se o iterativní optimalizaci nelineární váhové funkce a je definován následujícím vztahem:

$$\sum_{i=1}^{M}\sum_{j=1}^{N}\chi_{ij}(u_{ij}-P_iX_j)^T\sum{}_{ij}^{-1}(u_{ij}-P_iX_j), \tag{2.1}$$

kde P_i je matice kamery i-tého pohledu, X_j je j-tý bod v prostoru a $\sum_{ij}^{-1}(u_{ij}-P_iX_i)$ je kovarianční matice definující nepřesnost bodu u_{ij} [3]. Jejich součinem je získána zpětná projekce bodu na snímek. Při určování reprojekční chyby je důležitá celková chyba ze všech bodů.

2.2 Více-pohledová stereoskopie

Cílem více-pohledové stereoskopie je zahuštění řídkého mračna bodů získaného pomocí struktury z pohybu. Použitím struktury z pohybu jsou získány body v prostoru a matice kamer. Při znalosti těchto parametrů je možné určit větší množství bodů v prostoru a zahustit tak řídké mračno bodů. Při realizaci této metody se nejprve vyberou páry snímků jejíchž zorná pole se co nejvíce prolínají. Následně se pomocí nalezených párů vypočte hloubková mapa pro každý pohled, ze kterého je zaznamenána scéna. Získané hloubkové mapy se pro zpřesnění filtrují a jejich spojením vznikne zahuštěné mračno bodů.

2.2.1 Výběr párů

Prvním krokem více-pohledové stereoskopie je výběr párů pomocí kterých se budou počítat hloubkové mapy. Páry se vybírají pro každý pohled zaznamenávající scénu. Páry jsou tvořeny snímkem daného pohledu a snímkem zaznamenávajícím stejnou část scény. Snímky tvořící páry se snímkem daného pohledu jsou nazývány sousední snímky. Vhodným výběrem párů se zvýší přesnost získaných map a sníží výpočetní náročnost [26]. Vhodné páry by měly splňovat následující pravidla. Hodnota úhlu θ svíraného mezi zornými poli pohledů leží uvnitř intervalu definovaného vztahem 2.2.

$$5^\circ < \theta < 45^\circ \tag{2.2}$$

Dalším pravidlem je omezení vzdálenosti mezi kamerami pohledů. Délka báze b_i mezi pohledy není výrazně vychýlena vůči mediánu bází $\overline{b}$. Interval přípustného rozptylu

hodnot je vyjádřen ve vztahu 2.3 [26].

$$0,05\overline{b} < b_i < 2\overline{b} \tag{2.3}$$

Další možností, jak zpřesnit a optimalizovat výběr párů je omezení maximálního počtu sousedních snímků, které mohou být přiřazeny každému pohledu. Hodnota tohoto parametru se liší na základě barevné struktury scény, podle počtu pořízených snímků a podle způsobu jakým byly snímky pořízeny.

2.2.2 Výpočet hloubkových map

Pro každý pohled je pomocí párů definovaných v předchozí sekci vypočtena hloubková mapa. Hloubková mapa má stejný rozměr jako snímek zaznamenaný kamerou. Hodnoty jednotlivých pixelů hloubkové mapy odpovídají vzdálenosti daného bodu od kamery. Pro výpočet hloubkové mapy jednoho pohledu je potřeba mít alespoň jeden pár získaný postupem v předchozí sekci.

Hloubka bodů na snímcích je vypočtena pomocí triangulace. Pro možnost použití triangulace je nutná znalost korespondujících bodů daného páru. Detektory použité v této části hledají pro každý bod na referenčním snímku jeho korespondující bod na sousedním snímku. Detektory tímto způsobem naleznou větší množství bodů, než detektory použité při struktuře z pohybu (kap. 2.1), kde jsou korespondující body získány porovnáním příznakových vektorů významných bodů. Použité detektory se dělí na lokální a globální. Lokální detektory hledají korespondující bod pouze u sousedních pixelů, zatímco globální hledají body na celém sousedním snímku [9]. Pro lokální metody je problematická detekce ostrých změn a zakryté oblasti. Globální metody jsou výpočetně náročnější. Používané algoritmy jsou lokální metody Semi-global matching [10] a PatchMatch [11]. Originální algoritmy vyžadují rektifikované snímky, existují ale i alternativy, které lze aplikovat i na nerektifikované snímky [12][13].

V případě, že jsou snímky rektifikované je možné nejprve určit disparitní mapu. Disparitní mapa je snímek, ve kterém hodnotám jednotlivých pixelů odpovídá vzdálenost, o kterou je daný bod posunut na sousedním snímku. Body získané z rektifikovaných snímků jsou snáze detekovatelné, nicméně transformací v průběhu rektifikace zanikne část snímku a sníží se tak počet potenciálně detekovatelných bodů. Pro každý bod disparitní mapy, pro který se podařila určit disparita, je možné vypočítat hloubkovou mapu pomocí vztahu 1.13.

2.2.3 Spojení hloubkových map

V poslední části metody se spojí hloubkové mapy a vznikne tak výsledné mračno bodů. Před spojením map je nutné odstranit nadbytečné oblasti, které by způsobily chybné určení bodů v prostoru. Nadbytečné části jsou reprezentovány body na sousedních hloubkových mapách, které si ve skutečnosti neodpovídají. Mohou být způsobeny například vlastním stíněním objektu. Nadbytečné části se detekují zpětnou projekcí bodu z mapy jednoho pohledu do prostoru a následnou projekcí na mapy sousedních pohledů uplatněním vztahu 1.9. Bod sousedního pohledu je považován za nadbytečný a odebrán v případě, kdy má nižší hloubku než bod získaný projekcí. Pokud hloubka bodu získaného projekcí odpovídá hloubce bodu na mapě sousedního pohledu je bod také odebrán pro zabránění duplicity [13]. Tímto způsobem jsou odstraněny veškeré nadbytečné body na všech hloubkových mapách. Následně jsou všechny hloubkové mapy spojeny a zpětnou projekcí převedeny na body v prostoru.

2.3 Polygonální rekonstrukce povrchu

Výstupem z metod prostorové rekonstrukce (kap. 2.1 a kap. 2.2) je mračno bodů. Výsledný 3D model může být reprezentován pomocí získaných bodů, nicméně pro lepší vizuální vzhled a zaplnění prázdných oblastí je vhodné použití polygonální reprezentace. V této sekci budou popsány algoritmy, pomocí kterých je možné z mračna bodů vytvořit polygonální povrch, který bude reprezentovat výsledný 3D model.

2.3.1 Poissonova rekonstrukce povrchu

Metoda Poissonova rekonstrukce povrchu [14] řeší generování trojúhelníkové plochy z mračna orientovaných bodů jako Poissonův problém. Orientované body obsahují kromě trojrozměrných souřadnic také normály, které definují orientaci daného bodu. Cílem algoritmu je určit indikátorovou funkci, jejíž aproximací vznikne trojúhelníkový povrch. Indikátorová funkce nabývá hodnoty jedna uvnitř povrchu a hodnoty nula mimo povrch [15]. Orientované body je možné považovat za vzorky gradientu indikátorové funkce. Tyto body tvoří vektorové pole V, které reprezentuje gradientní pole indikátorové funkce. Aplikováním Poissonova vztahu (vztah. 2.4) je hledána hodnota indikátorové funkce χ s gradientem odpovídajícím V [15].

$$\Delta\chi = \nabla V \tag{2.4}$$

2.3.2 Delaunayova triangulace a tetrahedralizace

K vytvoření polygonálního povrchu lze využít Delaunayovu triangulaci. Pomocí tohoto algoritmu se z bodů vytvoří trojúhelníková síť. Body tvořící trojúhelníky musí splňovat následující pravidlo. Uvnitř kružnice procházející body vzniklého trojúhelníku neleží žádné body. Obdobou pro použití v 3D prostoru je Delaunayova tetrahedralizace. V tomto případě není vždy možné vytvořit síť trojúhelníků, jejichž plochy se neprotínají [16]. Řešením tohoto problému je přidání dodatečných vrcholů.

2.3.3 Optimalizace povrchu

Povrch získaný pomocí předchozích algoritmů nemusí plně odpovídat tvaru objektu. Důvodem může být šum obsažený v mračnu bodů, nebo způsob jakým byly vyplňovány prázdné oblasti. Tyto nepřesnosti lze do jisté míry redukovat optimalizacemi povrchu. K optimalizaci povrchu lze použít foto-konzistence, při které se vyšetřuje, zda jsou části povrchu viditelné z pohledů kamer. Optimalizace povrchu foto-konzistencí je možné docílit aplikováním algoritmu Graph-cut [17].

3 Mobilní platformy

V předchozí kapitole byl popsán způsob zpracovaní snímků pro vytvoření prostorového modelu. V této kapitole budou uvedeny platformy, na kterých je možné realizovat mobilní aplikaci, která bude používat zmíněné metody. Mobilní zařízení mají v dnešní době dostatečný výkon pro provádění složitých výpočtů a kvalitní fotoaparát potřebný pro zpracování obrazu. Jako mobilní zařízení jsou v rámci této práce myšleny mobilní telefony a tablety. Zařízení obsahují také snímače, které lze využít pro získání dodatečných informací sloužících k prostorové rekonstrukci. Tomuto tématu bude věnována kapitola 4.

Současnému trhu mobilních aplikací dominují platformy Android a iOS. Android používá 72,19 % uživatelů, zatímco iOS 26,99 % (duben 2021) [27]. Většina mobilních aplikací je tak cílena buď na jednu z těchto platforem, případně na obě. Každá z platforem má vlastní operační systém. Operační systémy mají rozdílné nativní programovací jazyky a aplikace mají rozdílnou strukturu. Přesto je možné vytvořit mobilní aplikaci, která bude dostupná pro více platforem. Tento typ aplikací se nazývá hybridní mobilní aplikace [28]. Aplikace je v tomto případě vyvíjena pro více platforem v rámci jednoho projektu se společným zdrojovým kódem a liší se pouze způsob jakým je sestaven instalační balíček. Hlavní výhodou hybridních aplikací je snadnější vývoj a údržba. Nevýhodou je nižší výkon aplikace a nižší dostupnost vývojových nástrojů oproti nativním aplikacím. Při zpracování obrazu je potřeba zajistit co nejlepší možný výkon a možnost použití potřebných nástrojů. Z toho důvodu se jeví jako vhodné vytvoření aplikace pouze pro jednu platformu, případně pro každou zvlášť. Použití hybridních aplikací by mělo význam v případě, kdy by výpočty probíhaly na cloudu a aplikace by pak sloužila k pořizování a zobrazování dat, případně pokud by bylo možné zajistit dostatečný výkon.

3.1 Význam mobilních platforem

Mobilní platformy s sebou přináší řadu výhod a nevýhod vůči ostatním platformám. Hlavní výhodou je kompaktnost a vysoká dostupnost mobilních zařízení. Není nutné realizovat měřicí systém složený z více modulů, protože jsou potřebné komponenty součástí zařízení. Při vývoji je nutné dodržovat pravidla pro vývoj na embedded platformách. Napsaný program by měl být efektivní vzhledem k omezenému výkonu a k minimalizaci spotřeby baterie. Podporovaná zařízení mohou mít rozdílné verze operačního systému, různé kvality snímačů a různé velikosti obrazovky. Aplikace by se měla chovat stejně pro různé podporované konfigurace. Požadované vlastnosti zařízení je možné definovat pomocí speciálních souborů Android manifest (kap. 3.2.1) a Info.plist (kap. 3.3.3).

3.2 Android

Majoritní postavení na trhu mobilních zařízení má platforma Android. Platforma Android má vlastní operační systém, jehož základem je jádro Linux [29]. Nativními jazyky jsou Kotlin a Java. Oficiální vývojové prostředí je Android Studio [30]. Android Studio je dostupné pro řadu operačních systémů a není povinnou součástí vývoje aplikace pro Android. Pro vývoj může být použito jiné dostupné vývojové prostředí a k sestavení výsledné aplikace lze použít sadu nástrojů Android SDK (Software Development Kit) [31]. Vývoj pro tuto platformu tak není omezen na specifický operační systém nebo software. Veškeré dostupné funkcionality operačního systému jsou dostupné pomocí obou nativních jazyků. Android také podporuje použití funkcí napsaných v jazycích C/C++ pomocí Android NDK (Native Development Kit) [32]. Je tak umožněno použít knihovny dostupné pro tyto jazyky, případně pomocí těchto jazyků implementovat algoritmy, které vyžadují co nejvyšší výkon. Kód napsaný v těchto jazycích je často nazýván jako nativní. Aplikace pro Android jsou primárně distribuovány v obchodě Google Play [33], nicméně existuje mnoho alternativ jako například Galaxy Store [34] a Amazon Appstore [35]. Výsledná aplikace je zkompilována do souboru APK (Android Package) [36], který obsahuje veškeré části aplikace potřebné k její instalaci na zařízení. Alternativou je AAB (Android App Bundle) [40] preferovaný při distribuci v obchodě Play. Jeho hlavní výhodou je menší velikost výsledné aplikace.

3.2.1 Struktura aplikace

Strukturu aplikace pro Android je možné rozdělit do tří vrstev. První z nich je Android Manifest obsahující informace o aplikaci. Komponenty definující chování aplikace a zdrojové soubory definující vzhled aplikace. Aplikace může být rozdělená do modulů. Každý z modulů má vlastní strukturu. Povinným modulem je samotná aplikace, nepovinným modulem jsou například knihovny.

Android Manifest

Každý projekt cílený pro platformu Android musí obsahovat Android Manifest [41]. Jedná se o *xml* soubor obsahující nezbytné informace o aplikaci. Mezi informacemi je zahrnut identifikátor aplikace, který je používaný k identifikaci v obchodě a jako jmenný prostor třídy projektu. Identifikátor je často nazýván jako název balíčku a konvence pro jeho tvar je *domena.spolecnost.nazevaplikace*, například *cz.vut.rekonstrukce*. V souboru jsou dále definovány použité komponenty a kompatibilní zařízení. Je tak možné definovat, že zařízení musí obsahovat požadovaný snímač, nebo určit minimální verzi operačního systému. V manifestu musí být uvedena

oprávnění, která aplikace vyžaduje pro přístup k zabezpečeným částem systému. Požadovaná oprávnění mohou být například přístup k fotoaparátu, nebo k souborům v úložišti zařízení. V Android Manifestu jsou dále definovány informace jako například orientace obrazovky, verze aplikace a ikony.

Komponenty

Komponenty definují chování aplikace. Mezi ně se řadí zdrojové soubory, které obsahují kód. Nejběžnější komponentou je aktivita. Aktivita slouží k interakci s uživatelem a reprezentuje obrazovku s uživatelským rozhraním. Dalšími komponenty jsou služby, sloužící pro operace běžící v pozadí. Poskytovatelé služeb, které lze použít pro uložení dat, například do databáze a přijímače vysílání pro přijímaní událostí od systému [36].

Zdrojové soubory

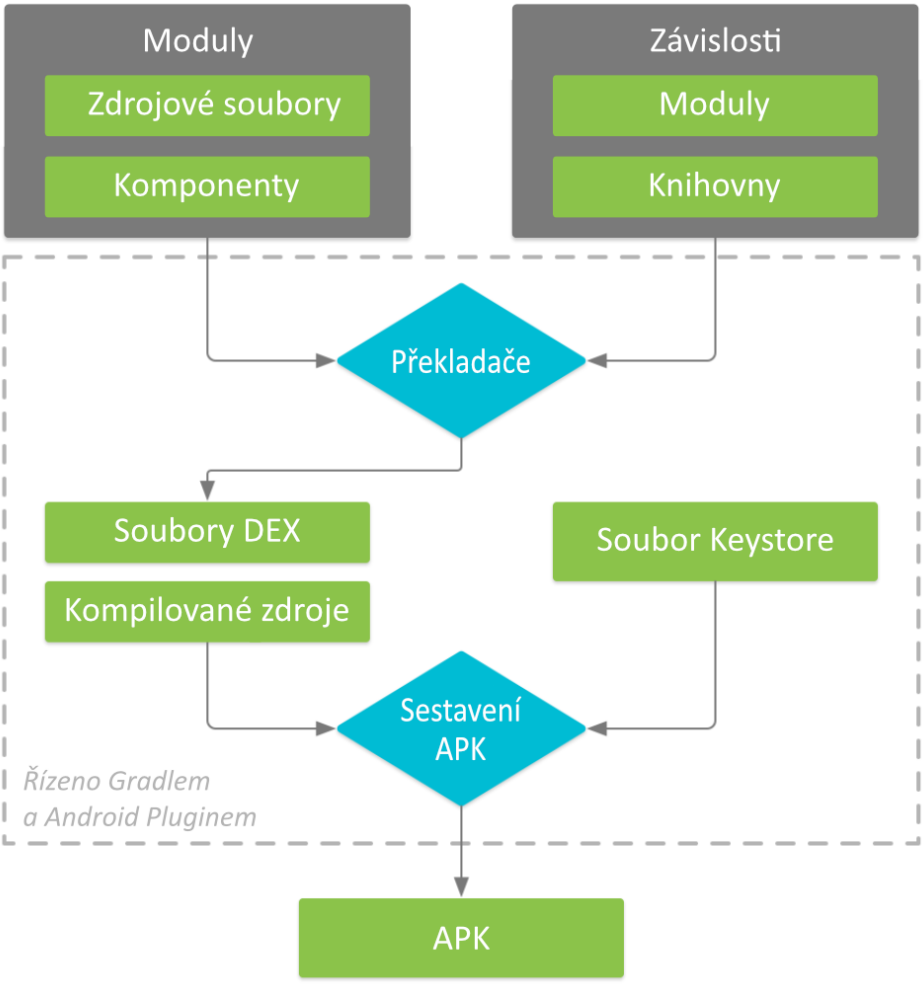
Zdrojové soubory zahrnují veškeré dodatečné soubory, které využívá kód aplikace. Jsou zde obsaženy definice rozložení obrazovek, ikony a další vizuální prvky. Oddělením těchto částí od kódu je zajištěna nezávislá údržba. Je také možné prakticky rozdělit zdrojové soubory pro různé konfigurace aplikace, například na základě rozměrů obrazovky, nebo jazyka [42].

3.2.2 Nástroje pro sestavení

K sestavení výsledné aplikace pro platformu Android slouží sada nástrojů Android SDK. K nástrojům lze přistoupit pomocí příkazové řádky, případně pomocí nástrojů pro automatizaci procesu sestavení. Android Studio používá k automatizaci sestavení Gradle [37], který umožňuje definovat rozdílná nastavení pro specifické konfigurace, aniž by byl modifikován zdrojový kód aplikace [38].

Proces sestavení

Proces sestavení slouží k vytvoření instalačního souboru APK ze zdrojových souborů projektu. Nejprve překladač konvertuje zdrojový kód do DEX (Dalvik Executable) souborů a části neobsahující kód do zkompilovaných zdrojových souborů [38]. Následně se sloučí DEX soubory a zkompilované zdrojové soubory do instalačního souboru APK, který je podepsán pomocí souboru *keystore*. Soubor *keystore* obsahuje certifikáty a klíče sloužící k ověření, že budoucí aktualizace pochází od původního autora [39].



Obr. 3.1: Proces sestavení aplikace pro Android (vychází z [38])

Konfigurační soubory

Konfigurační soubory slouží k definování pravidel, kterými se bude řídit Gradle při sestavení aplikace. Konfigurační soubory jsou definovány pro aplikaci a pro každý modul zvlášť. K popisu pravidel se používá jazyk DSL (Domain Specific Language). Na nejvyšší úrovni je *settings.gradle* který popisuje, které moduly jsou součástí projektu. Dále *build.gradle* definující konfigurace použité pro všechny moduly [38]. V tomto souboru jsou definovány závislosti jako například použité pluginy a knihovny. Každý modul má vlastní *build.gradle*, pomocí kterého jsou definovány specifické nastavení pro daný modul. Součástí je například verze SDK, která má být použita při sestavení a identifikátor aplikace, obdobně jako v Android Manifestu. V případě, že daný modul obsahuje kód napsaný v jazycích C/C++, je ke kompilaci těchto částí použit CMake [43]. CMake slouží k automatizaci překladu programů napsaných v nativních jazycích. Konfigurační soubory používají k definici pravidel vlastní

programovací jazyk CMake. Soubor *CMakeLists.txt* slouží ke konfiguraci a propojení nativního kódu s nativními knihovnami. Tento soubor musí být definován v *build.gradle* daného modulu.

3.3 iOS

Další platforma s dominantním postavením na trhu je iOS. Tato platforma má vlastní operační systém, který je dostupný pouze pro mobilní zařízení společnosti Apple. Nativním programovacím jazykem je Swift a oficiální vývojové prostředí je Xcode [44]. Operační systém také podporuje jazyky Objective-C a C/C++. Aplikace pro iOS jsou distribuovány pouze v obchodě App Store [45]. Na rozdíl od Android Studia je Xcode dostupný pouze pro operační systém macOS. V případě použití neoficiálního vývojového prostředí je přesto nutné provést kompilaci projektu v Xcode a nahrát aplikaci do App Store ze zařízení s macOS. Vývoj pro tuto platformu je tak omezen nejen počtem cílených zařízení, ale i počtem zařízení pomocí kterých lze aplikaci vyvíjet. Výhodou omezeného počtu cílených zařízení je přesná znalost použitého hardware, se kterou souvisí možnost optimalizace pro každé podporované zařízení. Výsledná aplikace je zkompilována do archivačního souboru IPA.

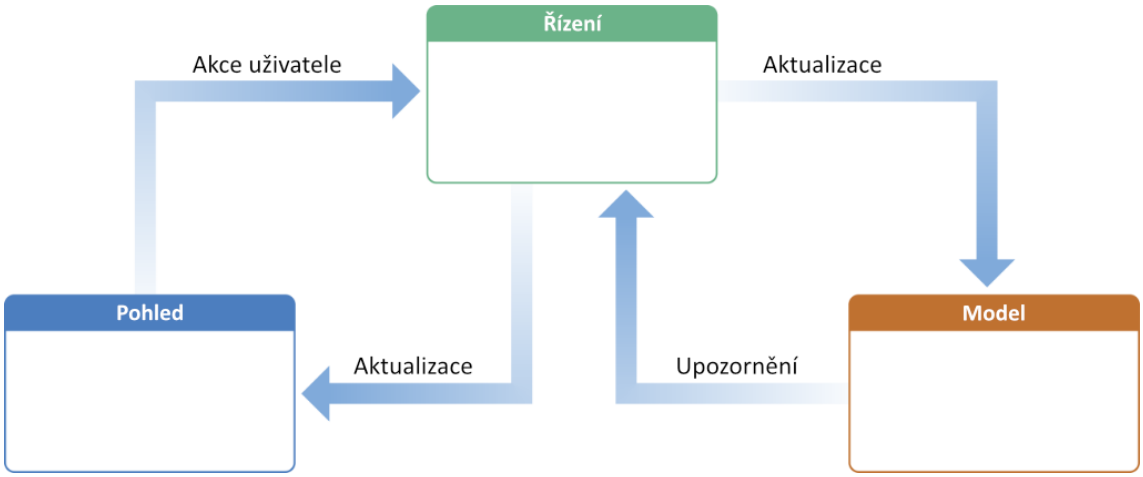
3.3.1 Architektura

Architektura iOS je složena ze čtyř vrstev. Vrstvy jsou Cocoa Touch, Media, Core Services a Core OS [46]. Každá z vrstev obsahuje řadu frameworků, které je možné použít při implementaci aplikace. Framework je hierarchický adresář obsahující sdílené zdrojové soubory, jedná se například o knihovny, hlavičkové soubory, dokumentaci a řetězce pro lokalizaci [47]. Výhodou frameworků je, že jsou veškeré části součástí jednoho balíčku. Většina frameworků je napsaných v Objective-C, je k nim ale možný přístup i ze Swiftu.

Vrstva Cocoa Touch zahrnuje technologie potřebné k vytvoření uživatelského rozhraní a pro řízení chování aplikace. Cocoa Touch obsahuje frameworky pomocí kterých je například možný přístup k událostem dotyku obrazovky, zpracování textu a přístup k souborům v úložišti. Vrstva media poskytuje grafické a mediální služby [48]. Jedná se například o renderování a animaci 2D a 3D grafiky, přehrávání zvuku a přehrávání videí. Core Services obsahuje frameworky pro práci se službami. Slouží například k určení polohy a pro přístup k datům ze snímačů [49]. Core OS slouží k poskytnutí služeb na nízké úrovni v souvislosti s hardware a sítí. Používá se například ke správě paměti, práci s vlákny a pro správu adresářů.

3.3.2 Návrhový vzor

Strukturu aplikací vyvíjených pro iOS definují návrhové vzory. Často používaný návrhový vzor pro aplikace iOS je Model-View-Controller. Vzor přidělí objektům v aplikaci jednu ze tří rolí: model, pohled (view) a řízení (controller) [50]. Tento vzor definuje roli objektu v aplikaci a způsob jakým mezi sebou komunikují. Použité objekty jsou opakovatelně použitelné. Aplikace používající tento návrhový vzor jsou snadno rozšiřitelné, vzhledem k tomu že většina frameworků Cocoa jsou vytvořeny na základě tohoto modelu a vyžadují, aby měly objekty některou ze zmíněných rolí.



Obr. 3.2: Návrhový vzor Model-View-Controller [50]

Model objektu obsahuje data specifická pro danou aplikaci a definuje logiku pro zpracování dat. Model pohledu představuje části, které jsou viditelné na obrazovce zařízení. Hlavním významem modelu pohledu je zobrazení dat z modelu objektu. Řídicí model se stará o chování modelů objektu a modelů pohledu. Slouží jako komunikace mezi modely, případně pro řízení úloh aplikace a spravování životního cyklu ostatních objektů [50].

3.3.3 Seznam vlastností

Seznam vlastností je obdobou dříve popsaného Android Manifestu pro zařízení iOS. Informace o aplikaci jsou zaznamenány v souboru *Info.plist* formou slovníku a přístup k hodnotám je možný pomocí páru klíč-hodnota [51]. Součástí je identifikátor balíčku, který má stejný tvar jako název balíčku v Android Manifestu. Nicméně k identifikaci v obchodě se používá unikátní App ID vygenerované v App Store. V

souboru jsou zahrnuty například požadované oprávnění, verze aplikace a kompatibilní zařízení. Z důvodu ochrany soukromí uživatele je nutné pro některá požadovaná oprávnění uvést z jakého důvodu je aplikace vyžaduje.

4 Získání dat

V předchozích kapitolách byl popsán způsob, jakým je možné docílit prostorové rekonstrukce pomocí snímků zachycujících stejnou scénu. Dále byly uvedeny vhodné mobilní platformy, pro které může být vytvořena aplikace využívající popsané metody. Pro správný průběh rekonstrukce je nutné získat potřebné informace o snímaném objektu. Tato kapitola se bude zabývat způsobem pořizování snímků pro zajištění kvalitního výstupu rekonstrukce. Dále budou uvedeny dodatečné informace o snímané scéně, které je možné získat pomocí mobilních zařízení.

V první části této kapitoly budou popsány parametry snímané scény a jaká pravidla by měla být dodržena pro pořízení snímků vedoucích k úspěšné rekonstrukci. V následující části kapitoly budou uvedeny způsoby, jakými je možné určit pozici a orientaci pohledů pomocí snímačů dostupných v mobilních zařízeních.

4.1 Definice scény

Vizuální parametry snímané scény mají vliv na kvalitu rekonstrukce. Hlavní vliv má barevná struktura snímané scény, její osvětlení a způsob jakým jsou pořízeny snímky. V této sekci budou uvedeny vhodné parametry scény a postupy při pořizování dat, které vedou k pořízení kvalitních dat potřebných k úspěšné rekonstrukci. Důležitým faktorem při pořizování dat je zachovaní barevné struktury scény na snímcích a zachování barevné konzistentnost mezi sousedními snímky pro maximalizaci detekovatelných bodů.

4.1.1 Barevná struktura

Důležitou vlastností snímané scény je její barevná struktura. Při použití metody struktura z pohybu jsou detekovány body ze snímků. Na objektech, které obsahují časté změny v barevné struktuře, rohy a hrany bude detekováno více bodů než na rovném jednobarevném povrchu. Na Obr. 4.1 je znázorněn vliv barevné struktury na počet detekovaných bodů. Na části snímku tvořené častými změnami barev a hranami je detekováno velké množství bodů, zatímco na jednobarevné stěně nejsou detekovány žádné body.

Obr. 4.1: Vliv barevné struktury na počet detekovaných bodů a) originální snímek, b) snímek s detekovanými body

4.1.2 Osvětlení

Dalším parametrem scény, který má vliv na kvalitu prostorové rekonstrukce je osvětlení. V ideálním případě by měly být všechny části objektu dostatečně a rovnoměrně nasvícené, aby se nelišily barvy stejných částí na snímcích zaznamenaných z jiných pozic. Osvětlení scény ovlivňuje i nastavení kamery. Kamera by měla mít vypnutý blesk a ostatní automatické vlastnosti upravující barevnou strukturu výsledného snímku [52]. Dále je vhodné dbát na to, aby se nevytvářely stíny na zaznamenávaných částech scény. Stíny vzniklé vlastním působením objektu je možné eliminovat přidáním dalšího zdroje osvětlení pod vhodným úhlem. Dalším praktickým využitím osvětlení může být nasvícení pouze částí scény, které mají být zaznamenané detailněji [53]. Intenzita osvětlení by neměla být ani příliš vysoká, aby nedošlo k znehodnocení barevné textury. Při návrhu osvětlení scény je nutné uvažovat i části objektu, které odráží, nebo případně vyzařují světlo. U venkovních scén je vhodné pořídit snímky v co nejkratším časovém úseku, vzhledem k časté změně podmínek. Změnu světelných podmínek je možné do jisté míry eliminovat vhodnou volbou detektoru při zpracování snímků.

4.1.3 Pořízení snímků

Způsob, jakým jsou pořízeny snímky sloužící k rekonstrukci, má vliv na přesnost výsledného modelu. Při pořizování snímků je nutné kromě dodržení požadavků na scénu popsaných v předchozích sekcích respektovat jistá pravidla. Pro snazší zpracování by měly být snímky stejné orientace a stejného rozlišení, což by neměl být v případě mobilní aplikace problém. Vzhledem k tomu, že budou na snímcích hledány korespondující body, by se měla překrývat dostatečná část snímků. Vzdálenost mezi pozicemi pořizovaných snímků by měla být co nejvyšší při které je zajištěno dostatečné překrytí a minimální ztráta viditelnosti bodů vlastním stíněním objektu.

4.2 Určení pozice a orientace

Mobilní zařízení jsou vybavena snímači, pomocí kterých lze měřit pozici a orientaci zařízení. Získáním těchto údajů pro každý pohled je zajištěna znalost vnějších parametrů při prostorové rekonstrukci. Přístup ke snímačům je u obou platforem obdobný. Zaregistrují se požadované snímače pomocí instancí příslušných objektů. U Androidu se jedná o *SensorManager* [54] a v případě iOS o *CMMotionManager* [55]. Následně jsou vykonávány události při každé nově naměřené hodnotě. Při použití snímačů je možné nastavit vlastní periodu vzorkování. Před použitím snímače je vhodné zkontrolovat, zda ho dané zařízení obsahuje.

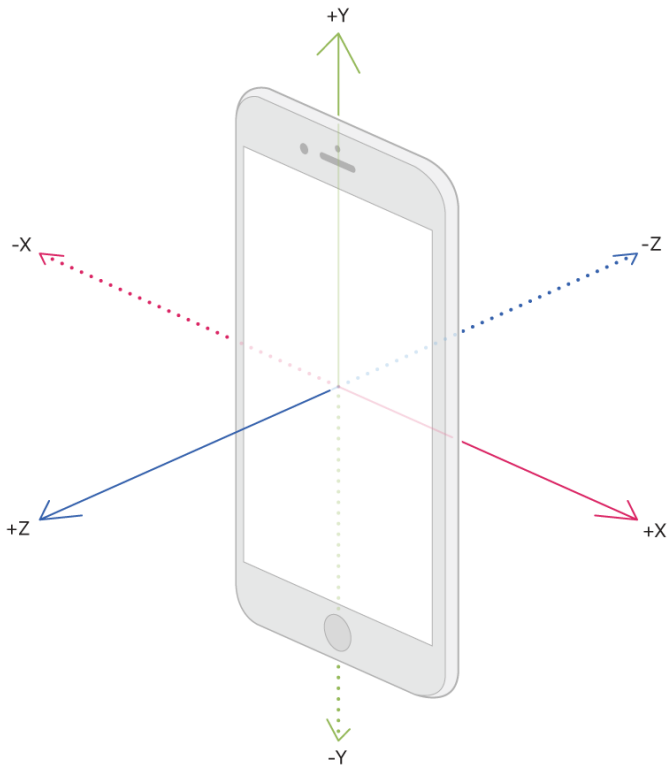
4.2.1 Měření pozice

Pozici je možné na mobilním zařízení měřit pomocí dvou způsobů. Způsoby se mezi sebou liší na základě měřené vzdálenosti. Pro delší vzdálenosti se používá GPS, zatímco pro kratší vzdálenosti je vhodné použití akcelerometru. Při měření vzdáleností mezi pozicemi, ze kterých jsou zaznamenány snímky potřebné k rekonstrukci scény je uvažováno měření kratších vzdáleností.

Akcelerometr je snímač sloužící k měření zrychlení. Je možné ho ale použít i pro nepřímé měření vzdálenosti vzhledem k fyzikálnímu vztahu mezi veličinami. Data získaná z akcelerometru jsou z pravidla doplněna naměřenými hodnotami ze snímače gravitace a případně z magnetometru [56]. Na zařízení působí zrychlení způsobené gravitací. Na naměřenou pozici by gravitace měla vliv neustálým přičítáním změny pozice a naměřená pozice by se nekontrolovaně měnila. Odečtením gravitace získané z tohoto snímače lze tento problém eliminovat.

Akcelerometr měří zrychlení pro každou ze tří os (obr. 4.2). Vzhledem k tomu, že akcelerometr měří zrychlení je potřeba naměřené hodnoty převést na změnu pozice pomocí dvojí integrace. Akcelerometr je charakteristický vysokým šumem, který

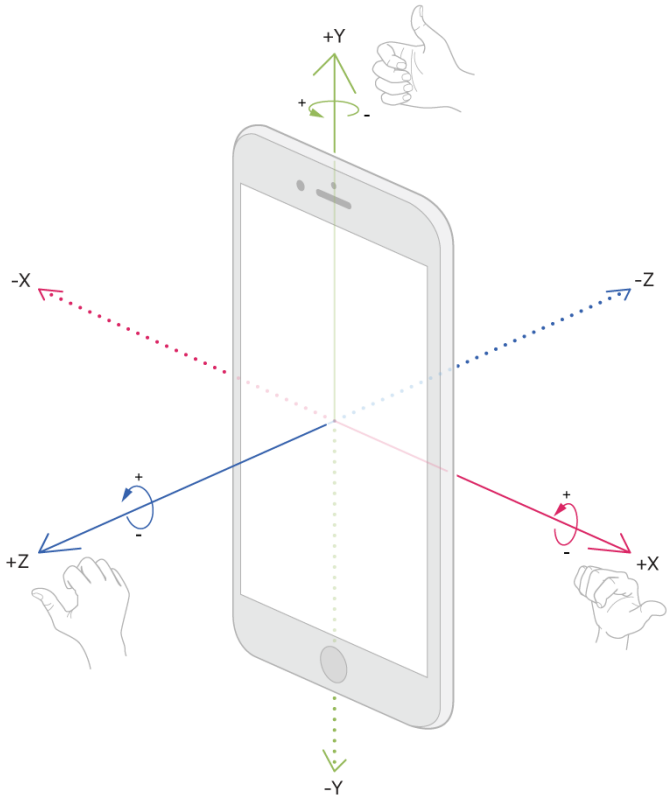
je po dvojí integraci značně zesílen. Ke zpřesnění dat z akcelerometru lze použít Kalmanův filtr [57].



Obr. 4.2: Směry pohybu měřené akcelerometrem [58]

4.2.2 Měření rotace

K měření rotace se používá gyroskop. Gyroskop měří úhlovou rychlost ve třech osách (obr. 4.3). Naměřené hodnoty gyroskopem mohou být zkresleny okolními vlivy, kterými může být například vlastní zrychlení zařízení [59]. Některé vlivy je možné sledovat pomocí snímačů gravitace a zrychlení [60]. Pro oba operační systémy jsou součástí rozhraní algoritmy, které naměřené hodnoty zpracují tak, aby neobsahovaly okolní vlivy a předají hodnotu rotace pro každou ze tří os (x, y, z). I u tohoto snímače je možné použití Kalmanova filtru pro zpřesnění naměřených dat.



Obr. 4.3: Úhlová rychlost měřená gyroskopem [59]

4.2.3 Použití hloubkového snímače

Řada nových zařízení obsahuje hloubkový snímač, který slouží k získání hloubkové mapy. Naměřené hloubkové mapy mohou být použity pro určení pozice a orientace zařízení, případně přímo k vytvoření prostorového modelu pomocí metody popsané v kapitole 2.2. V kombinaci s předchozími snímači se může jednat o dostatečně přesné měření k využití při prostorové rekonstrukci. Nicméně přesnost záleží na použitých snímačích a na tom, zda byly potřebné snímače na zařízení zkalibrovány výrobcem. Přístup k datům získaných tímto způsobem je možný pomocí knihoven rozšířené reality ARCore [61] a ARKit [62].

ARCore obsahuje Depth API, pomocí kterého lze získat hloubkovou mapu. ARCore je dostupný pro zařízení platforem Android a iOS. V současné době (květen 2021) je Depth API dostupné pouze pro omezené množství zařízení Android. Jedná se zejména o vlajkové lodě předních výrobců a malé množství zařízení střední třídy. Podporovaná zařízení musí mít certifikaci od Google. K získání certifikátu musí mít zařízení dostatečně výkoné CPU pro zajištění výpočtů v reálném čase a kvalitní potřebné snímače [64]. Kalibrované zařízení obsahují informace o fotoaparátu, ze kterých lze sestavit matici definující vnitřní parametry kamery [65].

ARKit je obdobou ARCore dostupný pouze pro zařízení iOS. ARKit obsahuje vlastní Depth API, které umožňuje získat hloubkovou mapu s využitím snímače LiDAR. Dostupnost tohoto snímače je požadovaná pro podporu Depth API. Snímač LiDAR je dostupný (květen 2021) pouze pro poslední generaci zařízení iOS [62].

4.2.4 Algoritmy zpracování obrazu

Další možností, jak získat pozici a orientaci je pomocí algoritmů zpracování obrazu. Tato varianta je často používána v rámci metody prostorové rekonstrukce struktura z pohybu (kap. 2.1). Použití tohoto způsobu je vhodné v případě, kdy není možné zajistit dostatečně přesné měření pozice a orientace. Princip spočívá v detekci korespondujících bodů, ze kterých se určí fundamentální matice, případně esenciální matice při znalosti vnitřních parametrů kamery. Ze zmíněných matic je následně možné určit vzájemnou změnu pozice a orientace mezi páry pohledů. Při implementaci lze využít pro korespondující body mezi dílčími pohledy data získaná při sledování vlastností (kap. 2.1.1).

5 Algoritmus rekonstrukce

V kapitole 2 byly uvedeny principy pomocí kterých lze vytvořit prostorový model ze snímků zachycujících snímanou scénu. Z nastíněných řešení byly po testech a úvahách zvoleny vhodné metody a algoritmy, které jsou popsány v této kapitole. Popsaný algoritmus bude použit v mobilní aplikaci. Vzhledem k tomu, že obě platformy popsané v kapitole 3 umožňují použití nativních funkcí napsaných v programovacích jazycích C/C++ je možné vytvořit algoritmus rekonstrukce nezávislý na cílové platformě. Převážně z důvodu znovupoužitelnosti kódu byl k implementaci zvolen právě jazyk C++. Dalším důvodem je zajištění co nejvyššího možného výkonu a možnost práce s pamětí kvůli cílení na hardware s omezeným výkonem a operační pamětí. Tento programovací jazyk také podporuje velké množství knihoven zpracování obrazu. Použité knihovny je nicméně nutné zkompilovat pro každou cílenou platformu a architekturu procesoru zvlášť. Algoritmus rekonstrukce byl nejprve implementován na počítači, pro možnost testování s nejvyšším možným výkonem a pro možnost ladění programu před tím, než bude realizována mobilní aplikace.

Samotný algoritmus rekonstrukce je rozdělen do následujících částí. V první části se detekují významné body na všech snímcích. Tyto body jsou porovnávány mezi snímky k vytvoření sledování vlastností. Následně se uplatní metoda struktura z pohybu pro získání řídkého mračna bodů a vnějších parametrů kamer. Druhou částí je zahuštění mračna bodů. Z pořízených snímků a dat z předchozí části se vytvoří hloubkové mapy pomocí kterých se získá více bodů v prostoru. Ze získaných bodů v prostoru se vygeneruje trojúhelníkový povrch, kterým je tvořen výsledný model. Na model je v závěru namapována textura získaná z původních snímků.

Program uvažuje situaci, kdy jsou k dispozici pouze snímky zaznamenané kamerou z více pozic. Algoritmus je navržen pro použití na mobilních zařízeních a je předpokládáno, že jsou snímky zachyceny pouze jednou kamerou, kterou představuje fotoaparát zařízení. Pro správný chod algoritmu je při pořizování dat nutné dodržet pravidla uvedená v kapitole 4.1. Algoritmus předpokládá, že nejsou k dispozici vnější, ani vnitřní parametry kamery. Důvodem je nízká přesnost snímačů pro určení pozice a orientace na mobilních zařízeních. Experimentálně bylo zjištěno, že ani při filtraci dat Kalmanovým filtrem není možné docílit dostatečné přesnosti. Použití hloubkového snímače nebylo uvažováno vzhledem k tomu, že je v současné době (květen 2021) podporován pouze na velmi omezeném počtu zařízení. Při použití vnitřních parametrů kamery v algoritmu je použit normalizovaný tvar.

Algoritmus rekonstrukce je realizován formou objektu, který se jmenuje *Reconstruction* a zahrnuje veškeré části procesu prostorové rekonstrukce. V objektu jsou k dispozici veřejné členské funkce pro předání dat potřebných k rekonstrukci. Funkce *AddImagePath* slouží k přidání absolutní cesty snímku, který má být použit při

rekonstrukci. *SetCachePath* slouží k nastavení adresáře pro ukládání dočasných souborů a pomocí *SetReconstructionParams* je možné nastavit parametry rekonstrukce. Tyto parametry budou popsány ve zbytku kapitoly u oblastí, na které mají vliv. Funkce *Reconstruct* slouží ke spuštění procesu prostorové rekonstrukce. Dílčí části algoritmu jsou popsány v následujících sekcích této kapitoly.

5.1 Realizace struktury z pohybu

V první části algoritmu rekonstrukce jsou zpracovány snímky pro získání bodů v prostoru a vnějších parametrů kamer. K získání těchto parametrů byla realizována metoda struktura z pohybu, která byla popsána v kapitole 2.1. V algoritmu je tato část implementována ve funkci *SparseReconstruction*. V této sekci budou popsány dílčí části, které jsou součástí této funkce.

5.1.1 Příprava snímků

V úvodní části algoritmu jsou načteny snímky z úložiště zařízení. K načtení dat byla implementována funkce *LoadImages*. Po načtení je zkontrolováno, zda jsou snímky v podporovaném formátu a zda jich je dostatečný počet. V případě, že je počet snímků nižší než tři, je program ukončen. Volitelně je sníženo rozlišení snímků pro snížení výpočetní náročnosti programu.

5.1.2 Detekce významných bodů

K detekci významných bodů slouží funkce *DetectKeyPoints*. V úvodní části funkce jsou detekovány významné body na snímcích pomocí detektoru AKAZE. Tento detektor byl zvolen z důvodu vysokého počtu reprodukovatelných bodů. Pro každý snímek jsou detekovány významné body. Každému detekovanému bodu je přiřazen příznakový vektor. Následně jsou vytvořeny páry ze všech možných kombinací snímků. V těchto párech jsou porovnávány příznakové vektory pro získání korespondujících bodů. Body musí splnit následující pravidla pro to, aby je bylo možné označit jako korespondující. Nejprve jsou pomocí algoritmu knnMatch nalezeny body druhého pohledu, které odpovídají bodům prvního pohledu. Body jsou ponechány pro další část pouze v případě, že je jejich vzdálenost v příznakovém prostoru nižší než předem definovaný práh. Následně je stejný postup aplikován v opačném pořadí. Druhý pohled je referenční a korespondující body se hledají na prvním. V další části se porovnají tyto získané body mezi sebou. Ponechají se pouze páry, které byly detekovány v obou případech. Závěrem je ze získaných bodů vypočítána fundamentální

matice pomocí algoritmu RANSAC [63]. Body jsou filtrovány pomocí epipolárního omezení (vztah 1.10).

5.1.3 Vytvoření sledování vlastností

Sledování vlastností je realizováno formou matice. Řádky matice představují snímky a sloupce představují nalezené vlastnosti. Prvky matice jsou souřadnice (u, v) dané vlastnosti na snímku. V případě že vlastnost není na snímku obsažena je souřadnicím přiřazena hodnota -1. Matice představující sledování vlastností má následující tvar:

$$\begin{array}{c} \\ I_1 \\ I_2 \\ \vdots \\ I_{n-1} \\ I_n \end{array} \begin{array}{c} \begin{array}{cccccc} F_1 & F_2 & \dots & F_{n-1} & F_n \end{array} \\ \begin{pmatrix} p_{1,1} & p_{1,2} & \dots & p_{1,n-1} & p_{1,n} \\ p_{2,1} & p_{2,2} & \dots & p_{2,n-1} & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{n-1,1} & p_{n-1,2} & \dots & p_{n-1,n-1} & p_{n-1,n} \\ p_{n,1} & p_{n,2} & \dots & p_{n,n-1} & p_{n,n} \end{pmatrix} \end{array}, \tag{5.1}$$

kde I_n je n-tý snímek, F_n je n-tá vlastnost a $p_{1,2}$ je vlastnost s indexem dva na prvním snímku. Vytvoření sledování vlastností je realizováno funkcí *CreateFeatureTracks*.

5.1.4 Rekonstrukce bodů

K rekonstrukci bodů v prostoru z detekovaných bodů na snímcích zaznamenaných pomocí sledování vlastností byl použit modul sfm [67] knihovny OpenCV. Modul obsahuje zjednodušenou verzi knihovny libmv [68] a implementaci triangulace včetně optimalizace pomocí algoritmu bundle adjustment. V průběhu rekonstrukce bodů je postupně pro páry obsahující dostatečný počet korespondujících bodů vypočtena esenciální matice pomocí osmi-bodého algoritmu. Z esenciální matice jsou určeny vnější parametry kamery, které jsou následně zpřesněny pomocí algoritmu bundle adjustment. Následně jsou pro dané páry vypočteny body v prostoru aplikací triangulace a pro zpřesnění získaných bodů je znovu použit bundle adjustment.

5.2 Zahuštění mračna bodů

Data získaná v předchozí části jsou použita k zahuštění mračna bodů. Zahuštění bodů je realizováno funkcí *DenseReconstruction*. Tato část algoritmu je vykonána pouze v případě, kdy byly získány vnější parametry všech pohledů. V opačném případě se pokračuje následující části pouze s řídkým mračnem bodů. K realizaci této

části byla použita knihovna OpenMVS [69], která obsahuje implementaci metody více-snímková stereoskopie (kap. 2.2).

5.2.1 Výpočet hloubkových map

První částí při zahuštění mračna bodů je výpočet hloubkových map. Před samotným výpočtem je i v tomto případě volitelně sníženo rozlišení snímků. Mobilní zařízení mají omezenou velikost operační paměti a při použití snímků v plném rozlišení by to v kombinaci s vypočtenými mapami mohlo vést k vyšším požadavkům na paměť, než by byla dostupná a tím i k neočekávanému pádu aplikace. Nejprve jsou hledány pro každý pohled sousední snímky, které zachycují stejnou část scény a je určeno v jaké části se sousední pohledy prolínají. Následně jsou pro každý pohled vypočteny hloubkové mapy pomocí algoritmu PatchMatch.

5.2.2 Spojení hloubkových map

Z vypočtených map jsou odfiltrovány body s příliš vysokou, nebo nízkou hloubkou. Z map jsou také odebrány nadbytečné oblasti popsané v kapitole 2.2.3. Filtrované hloubkové mapy jsou následně sloučeny do výsledného mračna bodů.

5.3 Polygonální rekonstrukce povrchu

K polygonální rekonstrukci povrchu slouží funkce *ReconstructMesh*. K vytvoření plochy z mračna bodů byla zvolena metoda Delaunayova tetrahedralizace, která je implementována v knihovně OpenMVS. Tato metoda byla zvolena z důvodu, že Poissonova rekonstrukce je příliš náchylná na spojení bodů, které nejsou součástí objektu. Dalším důvodem bylo, že tato knihovna již byla použita v předchozí části algoritmu a není tak nutné přidávat do algoritmu další závislosti.

Z mračna bodů je vygenerována trojúhelníková plocha pomocí Delaunayovy triangulace. Iterativně jsou propojeny nejbližší body tak, aby tvořily trojúhelníky. Bod je vložen do trojúhelníkové sítě pouze pokud je od napojovaného bodu ve vyšší vzdálenosti než předem definovaný práh. Povrch je následně optimalizován pomocí foto-konzistence. Závěrem je na povrch aplikována textura. Každé ploše je přiřazen snímek, ze kterého je daná plocha nejlépe viditelná. Ze snímků jsou získány barevné informace představující texturu. Získané textury ze snímků jsou uloženy na obrázek představující mapu na souřadnice, které odpovídají daným plochám.

5.4 Modularita algoritmu

Algoritmus rekonstrukce je vytvořen jako objekt, jehož části kódu nejsou závislé na specifické platformě. Tento způsob realizace byl zvolen, aby bylo možné kód sdílet mezi platformami beze změny kódu. Při současné realizaci se při použití na více platformách liší pouze způsob realizace mostu pro napojení na nativní jazyk platformy.

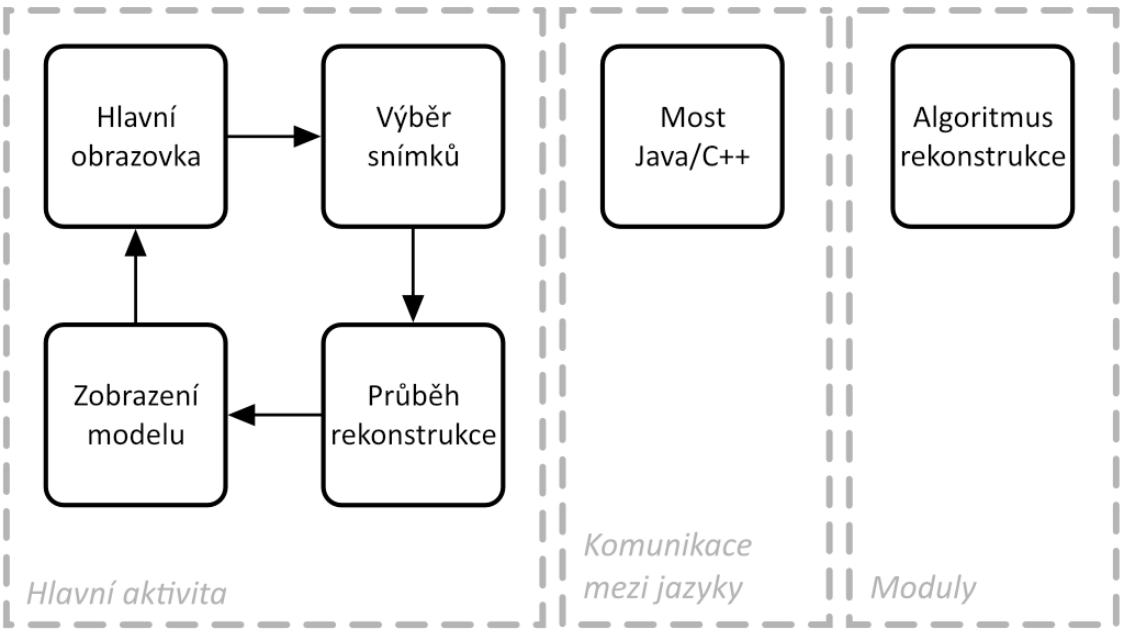
Algoritmus rekonstrukce komunikuje během vykonávání programu s nadřazeným modulem, který rekonstrukci spustil. V průběhu algoritmu dochází k validaci dat. V případě, že nejsou data validní, nebo dojde při vykonávání programu k chybě je algoritmus bezpečně ukončen a je předána chybová zpráva. Algoritmus také zaznamenává postup rekonstrukce a je umožněno předat algoritmu ukazatel na funkci, která bude volána při každé změně postupu.

6 Realizace mobilní aplikace

V předchozí kapitole byl popsán algoritmus prostorové rekonstrukce. Následující kapitola se bude zabývat realizací mobilní aplikace, která bude používat tento algoritmus. Pro realizaci aplikace jsou vhodné obě zmíněné platformy v kapitole 3. Z uvedených platforem byla zvolena platforma Android z důvodu vyššího zastoupení počtu zařízení na trhu. Pro vývoj aplikace bylo zvoleno vývojové prostředí Android Studio. Jedná se o oficiální vývojové prostředí a jsou tak k dispozici veškeré potřebné nástroje pro vývoj a ladění programu. Pro implementaci aplikace byl zvolen programovací jazyk Java.

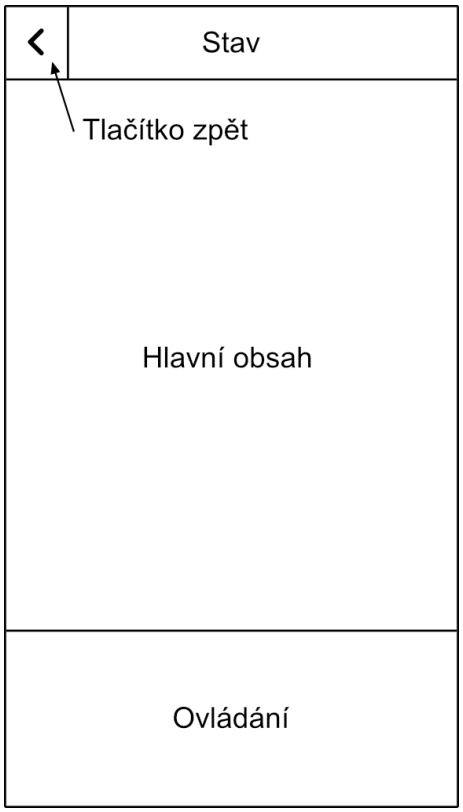
6.1 Struktura aplikace

Aplikace je rozdělena do logických celků, které představují obrazovky aplikace. Strukturu aplikace tvoří hlavní obrazovka, výběr snímků sloužících k rekonstrukci, obrazovka popisující průběh rekonstrukce a zobrazení výsledného modelu (Obr. 6.1). Tyto části jsou obsaženy v hlavní aktivitě, která je implementována v souboru *MainActivity.java*. Součástí je také algoritmus rekonstrukce, který je z pohledu aplikace brán jako samostatný modul a je implementován v souborech *Reconstruction.cpp* a *Reconstruction.h*. Pro komunikaci mezi hlavní aktivitou a algoritmem rekonstrukce slouží most, který je implemenován v souboru *Bridge.cpp*.



Obr. 6.1: Struktura aplikace

Rozložení vrstev na obrazovce bylo navrženo tak, aby bylo dodrženo preferované rozložení pro danou platformu a aby měla aplikace jednoduché a očekávané ovládání. Rozložení obrazovky je totožné v rámci celé aplikace. Mezi obrazovkami se liší pouze obsah daných vrstev. Rozložení aplikace je zobrazeno na Obr. 6.2. Ve vrchní části obrazovky je zobrazen aktuální stav aplikace. V levém horním rohu je tlačítko pro návrat na předchozí obrazovku. Ve spodní části je ovládání, jedná se převážně o tlačítka pro navigaci na jiné obrazovky. Zbylou část obrazovky tvoří hlavní obsah, který je charakteristický pro danou obrazovku. Může se jednat například o renderování 3D modelu, nebo zobrazení snímků. Rozložení obrazovek je definováno v souboru *activity_main.xml*.

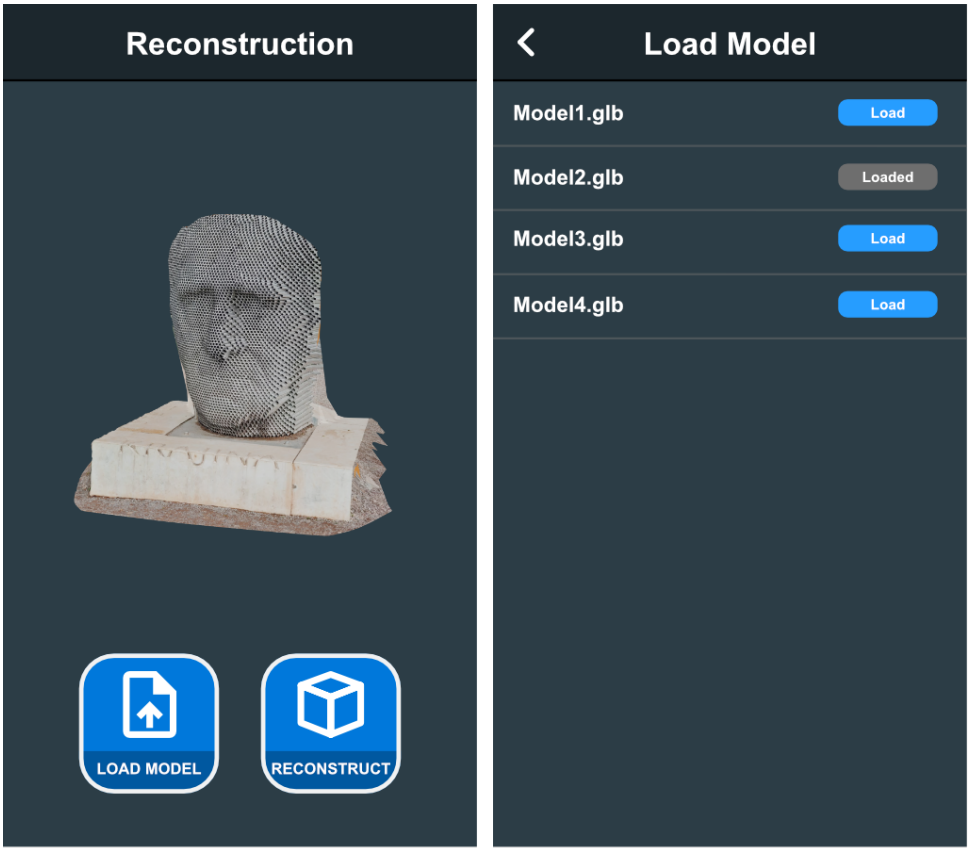


Obr. 6.2: Rozložení aplikace

6.1.1 Hlavní obrazovka

Vstupním bodem aplikace je hlavní obrazovka (Obr. 6.3a). Tato obrazovka slouží k navigaci k ostatním obrazovkám. Hlavní obsah obrazovky je tvořen 3D modelem. Pro renderování 3D modelu byla použita knihovna Sceneform [70], která podporuje formáty glb a gltf. Z této obrazovky je možné pomocí tlačítek načíst model z úložiště zařízení, nebo začít proces rekonstrukce, který zobrazí obrazovku s načítáním

snímků. Při kliknutí na tlačítko pro načtení modelu jsou nalezeny veškeré podporované 3D modely v zařízení a jsou vypsány na obrazovku (Obr. 6.3b). Seznam modelů v zařízení je vypsán pomocí dynamického seznamu RecyclerView [71], který je implementován v souboru *ModelPick.java*. Po výběru modelu se aplikace vrátí na hlavní obrazovku a je zobrazen vybraný model. Kliknutím na tlačítko rekonstrukce je otevřena obrazovka pro výběr snímků.

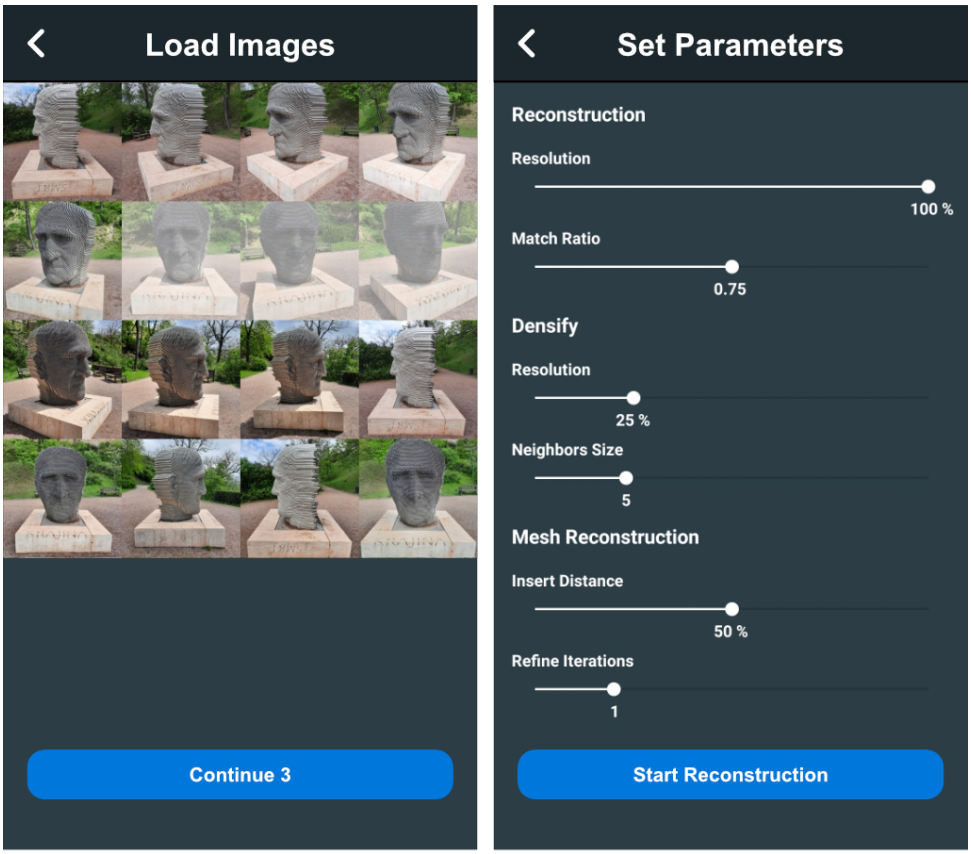


Obr. 6.3: a) Hlavní obrazovka, b) Načtení modelu

6.1.2 Výběr snímků

První částí procesu rekonstrukce je výběr snímků. Obrazovka pro výběr snímků (Obr. 6.4a) zobrazí náhled obrázků, které se nachází v úložišti zařízení. Pro zobrazení náhledu obrázků je i v tomto případě použit RecyclerView, který je implementován v souboru *ImagePick.java*. Náhledy obrázků jsou interaktivní a dotykem na jednotlivé položky jsou vybrány snímky, které budou použity při rekonstrukci. Při výběru jsou zaznamenány absolutní cesty vybraných snímků, které jsou po potvrzení předány algoritmu rekonstrukce. Snímky jsou tak načteny až ve chvíli, kdy jsou potřebné pro zpracování. Výběr snímku je možné zrušit opětovným kliknutím

na náhled. Ve spodní části je po označení dostatečného počtu snímků zobrazeno tlačítko pro potvrzení. Po jeho stisknutí se zobrazí obrazovka pro nastavení parametrů rekonstrukce.



Obr. 6.4: a) Výběr snímků, b) Nastavení parametrů rekonstrukce

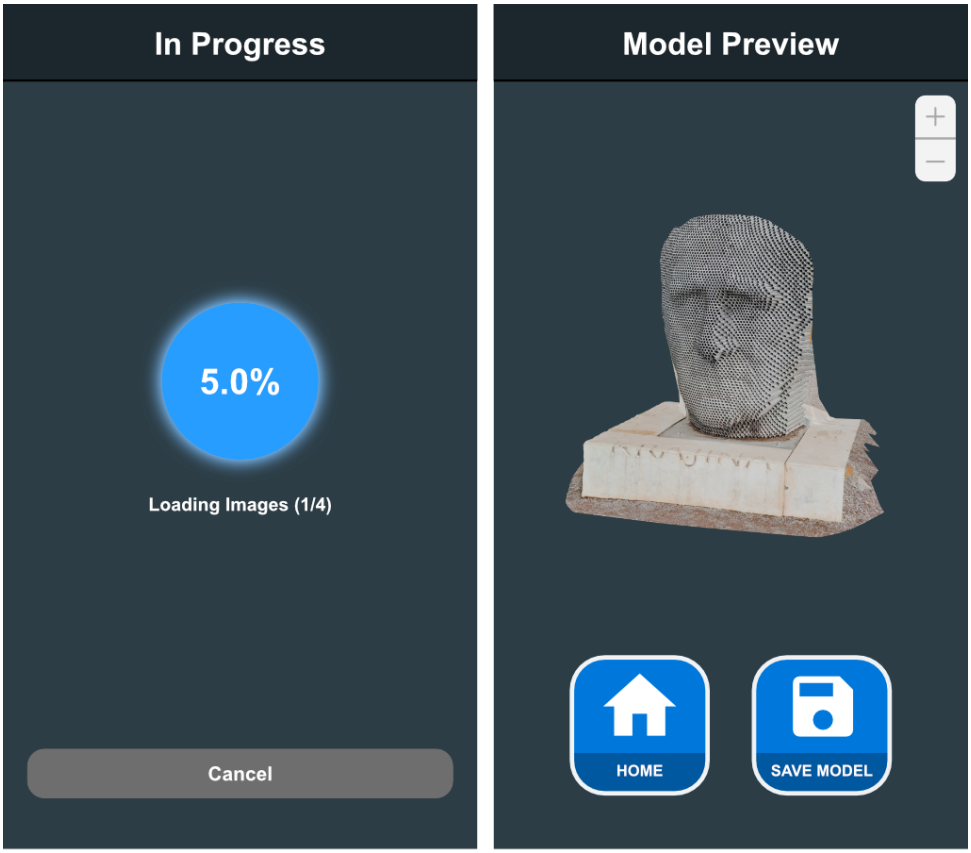
6.1.3 Nastavení parametrů rekonstrukce

Před samotným spuštěním algoritmu rekonstrukce je uživateli umožněno nastavit parametry pro ovlivnění průběhu rekonstrukce. K nastavení těchto parametrů slouží obrazovka zobrazená na Obr. 6.4b. Jedná se o parametry, které byly zmíněny při popisu algoritmu rekonstrukce v kapitole 5 jako volitelné. Parametry jsou přednastaveny na doporučené hodnoty a jejich změna je možná pomocí jezdce. Parametry jsou rozděleny do tří sekcí, na základě toho, na kterou část algoritmu mají vliv. V rámci získávání mračna bodů je možné nastavit rozlišení snímků a hodnotu prahu pro filtraci korespondujících bodů. Pro druhou část algoritmu, při které je zahuštěno mračno bodů lze nastavit relativní hodnotu rozlišení snímků vůči hodnotě rozlišení použité v předchozí části a maximální počet sousedních snímků každého pohledu. Pro poslední část algoritmu, která slouží k rekonstrukci polygonálního povrchu je

umožněno nastavit maximální přípustnou vzdálenost mezi propojenými body při generování trojúhelníků a počet iterací při optimalizaci povrchu. Po stisknutí tlačítka ve spodní části obrazovky je spuštěn algoritmus rekonstrukce a zobrazena obrazovka s průběhem rekonstrukce.

6.1.4 Průběh rekonstrukce

Během vykonávání algoritmu rekonstrukce je aktivní obrazovka zobrazující postup rekonstrukce (Obr. 6.5a). Na obrazovce je zobrazen aktuální postup v procentech a je vypsáno v jaké fázi se algoritmus právě nachází. Průběh rekonstrukce je doprovázen animací, při které je vyzařován kulatý vzor. Důvodem je, aby obrazovka nepůsobila staticky a nebudila tak dojem, že se aplikace zasekla. Rekonstrukci je možné kdykoliv ukončit tlačítkem pro ukončení rekonstrukce. Při jeho stisknutí je zobrazena hlavní obrazovka. Při úspěšném dokončení rekonstrukce se zobrazí obrazovka pro zobrazení modelu. V případě, že rekonstrukce není úspěšná se zobrazí chybová zpráva předaná algoritmem rekonstrukce a aplikace se vrátí na hlavní obrazovku.



Obr. 6.5: a) Postup rekonstrukce, b) Zobrazení modelu

6.1.5 Zobrazení modelu

Pro zobrazení výsledků získaných algoritmem rekonstrukce slouží obrazovka pro zobrazení modelu (Obr. 6.5b). Přiblížení a oddálení modelu je umožněno pomocí tlačítek v pravém horním rohu obrazovky. Rotace modelu lze docílit tahem po obrazovce. Při stisknutí tlačítka pro uložení modelu, se model uloží ve formátu glb do úložiště zařízení. Z této obrazovky je možný také návrat na domácí obrazovku, ze které lze proces rekonstrukce opakovat.

6.2 Připojení závislostí

Aplikace používá řadu závislostí, které bylo nutné připojit k aplikaci a případně i zvlášť zkompilovat. Závislosti, které jsou použity v hlavní aktivitě aplikace jsou pouze uvedeny v souboru *build.gradle* aplikace a o jejich napojení se postará Gradle při sestavení aplikace. Jedná se o části kódu, který je napsán v Javě. Jsou to například knihovny použité při zobrazení modelu, nebo náhledu snímků. V případě knihoven použitých v algoritmu rekonstrukce bylo nutné tyto knihovny nejprve zkompilovat a poté je k aplikaci připojit. V tomto případě se jedná o knihovny použité v části programu napsaném v jazyce C++ a jsou to převážně knihovny zpracování obrazu. Tyto knihovny byly sestaveny včetně veškerých jejich závislostí pomocí CMake. K sestavení byly napsány Shell skripty. Výsledné sdílené, případně statické knihovny jsou cílené na platformu Android a architekturu arm64-v8a. Sestavení pro jinou architekturu je umožněno změnou parametru ve skriptu. Sestavené knihovny jsou k aplikaci připojeny v souboru *CMakeLists.txt*.

Knihovny, které bylo nutné sestavit jsou OpenCV a OpenMVS. Základní verze knihovny OpenCV je k dispozici pro platformu Android, nicméně v algoritmu rekonstrukce byl použit modul sfm, který není součástí základní verze. Z toho důvodu bylo nutné zkompilovat tuto knihovnu ze zdrojových souborů. Modul sfm je závislý na knihovnách Eigen [72], glog [73], gflags [74] a Ceres Solver [75]. Knihovnu Eigen nebylo nutné kompilovat, při použití se pouze vkládají její zdrojové soubory. Pro zbylé závislosti byly napsány skripty, pomocí kterých byly knihovny sestaveny. U knihoven gflags a Ceres musel být nastaven režim pro práci pouze na jednom vlákně. Důvodem je, že Android NDK neobsahuje některé použité funkce pro práci s vlákny. Bylo také nutné použít verze knihoven, které jsou navzájem kompatibilní, případně upravit skripty CMake pro hledání závislostí knihoven. Skripty pro hledání závislostí uvažovaly rozdílnou strukturu zdrojových souborů knihovny, nebo jmenný prostor knihovny starší verze. Poté byla zkompilována knihovna OpenCV obsahující modul sfm. Při překladu byly připojeny veškeré závislosti a bylo určeno, které moduly mají být zkompilovány a které ne pro minimalizaci velikosti výsledné

knihovny.

Knihovna OpenMVS požaduje řadu závislostí. Část z nich již byla zkompilována při překladu knihovny OpenCV a bylo je tak možné použít znovu. Kromě těchto knihoven byly požadovány závislosti CGAL [76], Boost [77] a VCG [78]. Tyto knihovny byly také nejprve přeloženy zvlášť včetně jejich závislostí. Problematickou částí při kompilaci knihovny OpenMVS bylo, že zdrojový kód byl původně cílen na operační systém Windows. Knihovna sice byla rozšířena o podporu dalších operačních systémů, nicméně přesto byly ve zdrojovém kódu použity příkazy specifické pro danou platformu. Tyto části bylo nutné postupně změnit na příkazy podporované Android NDK. Poté byly napojeny na knihovnu veškeré závislosti a knihovna byla zkompilována.

6.3 Cílená zařízení

Minimální verze operačního systému, kterou musí zařízení mít je Android 8.0. Jedná se o nejnižší verzi, která obsahuje použité funkce a je vyžadována použitými závislostmi. Aplikace vyžaduje oprávnění, pro které je nutný souhlas uživatele. Jedná se o oprávnění pro čtení a zápis do úložiště zařízení. Oprávnění jsou vyžádána až ve chvíli, kdy je aplikace potřebuje. Oprávnění je potřebné pro načtení snímků a pro uložení výsledného modelu. Aplikace byla sestavena pro architekturu arm64-v8a, která je momentálně (květen 2021) standardem pro zařízení Android. Instalační soubor APK je podepsán pomocí souboru *keystore* a aplikace tak splňuje náležitosti pro publikování v obchodě Play.

7 Testování

V předchozí kapitole byla popsána aplikace sloužící k prostorové rekonstrukci. V této kapitole bude aplikace otestována na mobilním zařízení. Mobilní aplikace pro 3D rekonstrukci byla otestována rekonstrukcí objektů a tím i její reálné využití. Pro otestování metod byly vytvořeny modely dvou objektů. Bubnu, při kterém se testovala rekonstrukce ve vnitřních podmínkách a sochy, při které byly pořízeny snímky ve venkovních podmínkách. Při rekonstrukci jsou vytvářeny komplexní modely, u kterých je obtížné numericky vyjádřit přesnost. Pro zhodnocení přesnosti by bylo nutné mít k dispozici referenční 3D model, který lze získat například laserovým měřením. Získané modely jsou tak hodnoceny převážně z vizuálního hlediska porovnáním se snímky, které zachycují scénu.

Testování aplikace proběhlo na zařízení Samsung Galaxy A51. Jedná se o zařízení střední třídy a také o celosvětově nejprodávanější zařízení platformy Android pro první čtvrtletí roku 2020 [79]. Jedná se tak o vhodný výchozí bod pro testování. Zařízení má výkonný osmi jádrový procesor s kmitočtem 2,3 GHz pro čtyři jádra a s kmitočtem 1,7 GHz pro zbylé čtyři jádra. Verze zařízení prodávaná v České republice disponuje operační pamětí o velikosti 4 GB, což může být při vykonávání algoritmu rekonstrukce stěžejním, vzhledem k vysokým nárokům na operační paměť při generování hloubkových map. Je nutné brát v úvahu, že aplikace nemá za běhu k dispozici velké množství z celkové velikosti operační paměti, protože část využívá systém, procesy běžící v pozadí a část nevyužité paměti je rezervována systémem. Z toho důvodu bylo při testování rekonstrukce nutno omezit počet a rozlišení zpracovaných snímků.

Při pořizování dat byly dodrženy podmínky popsané v kapitole 4. Bylo dbáno na dodržení barevné struktury objektu na snímcích mezi pohledy. Vzhledem k tomu, že algoritmus využívá k výpočtu vnějších parametrů detekované body na snímcích, byly pozice, ze kterých byly pořízeny data zvoleny tak, aby se zorná pole sousedních snímků překrývala alespoň o 80 %. Zmíněné pravidlo sice pomohlo ke zvýšení počtu detekovaných bodů, ale byla tím i snížena velikost báze mezi sousedními snímky.

7.1 Rekonstrukce bubnu

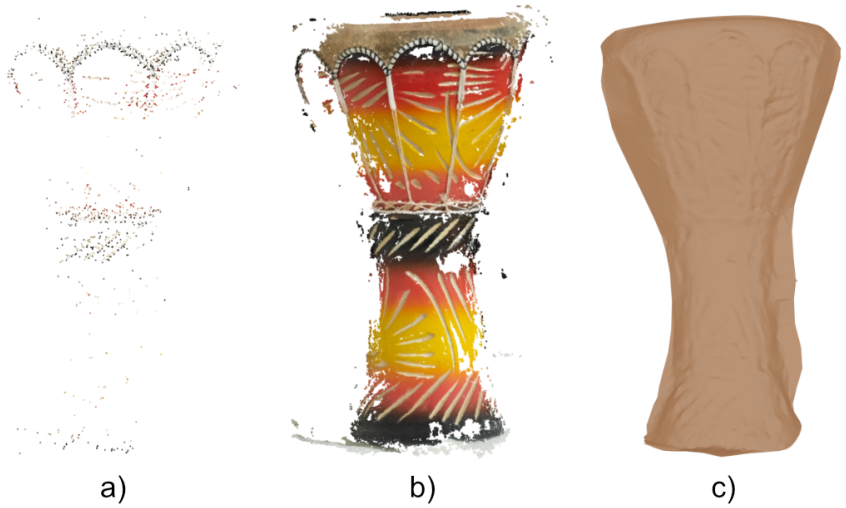
Prvním objektem pro otestování aplikace je buben (Obr. 7.1). Tento objekt byl zvolen z důvodu vysokého obsahu barevné textury a vyřezaných vzorů po obvodu objektu. Objekt tak obsahuje velké množství potenciálně detekovatelných bodů. Při testování tohoto objektu byla uvažována situace, při které jsou pořízeny snímky ve vnitřních podmínkách. Je tak možné přizpůsobit světelné podmínky a okolí měřené scény. Při pořizování dat bylo za objektem vyvěšeno černé plátno, pro redukci

detekovaných okolních bodů a tím i ke zrychlení zpracování dat.

Obr. 7.1: Buben

Z důvodu omezené operační paměti testovaného zařízení nebyly zpracovány snímky po celém obvodu objektu, ale pouze z přední části. Vzhledem k tomu, že je tvar objektu po celém obvodu shodný, se jedná o dostatečný způsob, jak otestovat navržené algoritmy. Pro rekonstrukci objektu byly pořízeny 4 snímky. Rozlišení zpracovaných snímků bylo sníženo na 50 %. Rozlišení zpracovaných snímků tak bylo 1 500 x 2 000 px. Pokud by se snímky zpracovaly v plném rozlišení by jejich počet musel být nižší. Ze zpracovaných snímků bylo získáno řídké mračno bodů (Obr. 7.2a) složené z 1 887 bodů v prostoru. Tyto body byly použity k zahuštění mračna bodů (Obr. 7.2b). Pro zpracování snímků v této části algoritmu bylo rozlišení snímků znovu sníženo na 50 %. Počet bodů v prostoru se zvýšil na 201 379 bodů. Následně byla vytvořena polygonální plocha (Obr. 7.2c) a byla aplikována textura získaná z barevných informací na snímcích.

Výsledný model (Obr. 7.3) odpovídá rekonstruovanému objektu. Při rekonstrukci se podařil zachytit tvar objektu i hloubka vzorů které jsou viditelné po obvodu objektu. Textura dosahuje vysoké kvality, vzhledem k tomu že bylo možné přizpůsobit osvětlení scény. Jedinou nevýhodou modelu jsou chybějící oblasti, které nebyly zachyceny na zpracovaných snímcích. Získání úplného modelu by bylo možné při zpracování dat na zařízení s vyšší operační pamětí. Další možností by byla úprava



Obr. 7.2: Rekonstrukce bubnu a) Řídké mračno bodů, b) Zahuštěné mračno bodů, c) Polygonální plocha

algoritmu tak, aby nebylo nutné určovat vnější parametry kamer z detekovaných bodů na snímcích. Báze mezi sousedními snímky by tak mohla být vyšší a omezený počet zpracovatelných snímků by tak byl dostatečný pro zachycení veškerých částí objektu. Musela by být zvolena dostatečně přesná alternativa získání vnějších parametrů. Další možností je optimalizace algoritmu pro snížení nároků na paměť. Na okrajích výsledného modelu jsou viditelné mírné přesahy, které neodpovídají geometrii objektu. Tyto přesahy vznikly při generování polygonového povrchu spojením bodů, které ve skutečnosti neodpovídají objektu. Řešením tohoto problému by mohlo být zpracování více pohledů.

Obr. 7.3: Model bubnu

7.2 Rekonstrukce sochy

Druhým objektem, na kterém byla otestována aplikace je socha (Obr. 7.4). Tento objekt byl zvolen pro otestování ve venkovních podmínkách, ve kterých jsou omezené možnosti pro přizpůsobení scény. Objekt obsahuje velké množství hran, které jsou detekovatelné algoritmy a je tak vhodný pro otestování algoritmu. Pro pořízení snímků byla zvolena denní doba, při které nedochází k výrazným změnám v osvětlení. Přesto má objekt rozdílnou barevnou strukturu pro odlišné pohledy. Z tohoto důvodu s kombinací s omezeným množstvím snímků, které dokáže zvolené zařízení pro testování zpracovat, byl model sochy vytvořen z přední a bočních stran zvlášť. Při rekonstrukci byly zachyceny i oblasti, které nepředstavují testovaný objekt. Jedná se například o krajinu, která je v pozadí sochy. Tyto oblasti byly z výsledných modelů odebrány.

Obr. 7.4: Socha

7.2.1 Přední pohled

Pro rekonstrukci přední strany sochy byly pořízeny čtyři snímky zachycující tuto oblast. Rozlišení snímků bylo pro získání řídkého mračna bodů sníženo na 40 % na 1 200 x 1 600 px. Zpracováním těchto snímků bylo získáno 2 743 bodů v prostoru (Obr. 7.5a). Při zahuštění bodů bylo rozlišení snímků sníženo o dalších 50 % a mračno bodů bylo zahuštěno na celkový počet 238 750 bodů. (Obr. 7.5b). Z bodů byla vygenerována polygonální plocha (Obr. 7.5c). Pro její vytvoření nebyly změněny

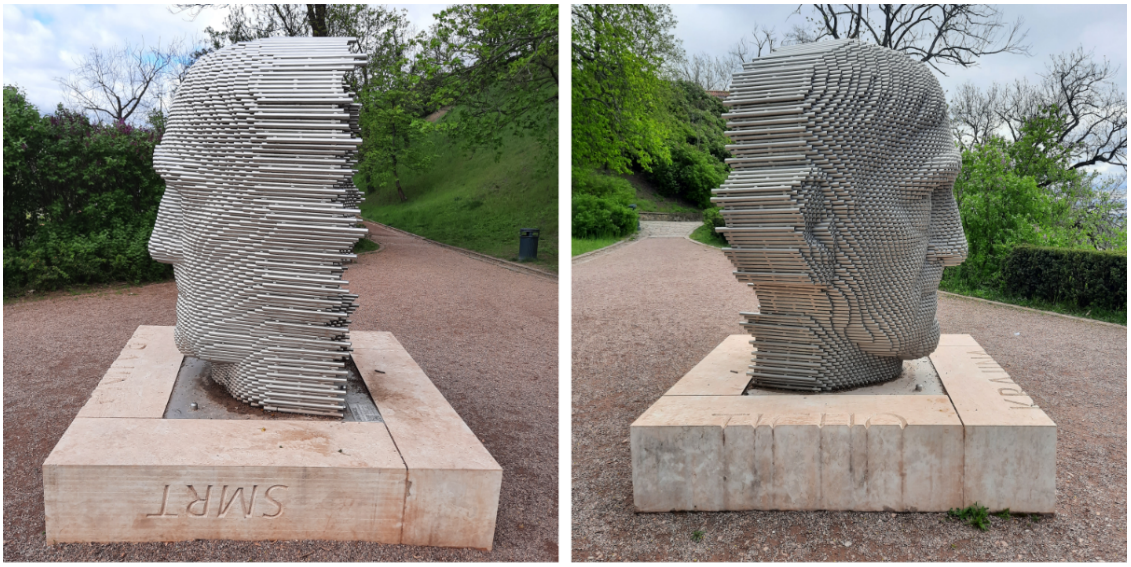
parametry. Výsledný model, na kterém je namapována textura ze snímků je zobrazen na Obr. 7.6.

Obr. 7.5: Rekonstrukce sochy zepředu a) Řídké mračno bodů, b) Zahuštěné mračno bodů, c) Polygonální plocha

Výsledný model vizuálně odpovídá rekonstruovanému objektu. Textura na modelu má místy tmavší odstín. Důvodem je vlastní stínění objektu. Pro řešení tohoto problému by bylo nutné přizpůsobit osvětlení scény.

Obr. 7.6: Přední pohled modelu sochy

Hlavní předností tohoto modelu je, že se podařilo zachytit hloubku detailních oblastí, jako například nápis na přední části podstavy sochy. Další předností je vyplnění oblastí, na kterých bylo obtížné detekovat body.



a) b)

Obr. 7.7: Pohled na sochu z boku a) Pravá strana, b) Levá strana

7.2.2 Levá strana

Pro rekonstrukci levé strany sochy (Obr. 7.7b) byly pořízeny čtyři snímky zachycující tuto oblast. Rozlišení snímků bylo sníženo na 50 % na 1 500 x 2 000 px. Řídké mračno bodů je složeno z 3 117 bodů (Obr. 7.8a). Tyto body byly následně zahuštěny na celkový počet 204 365 bodů (Obr. 7.8b). Model získaný polygonální rekonstrukcí povrchu je zobrazen na Obr. 7.8c.

a) b) c)

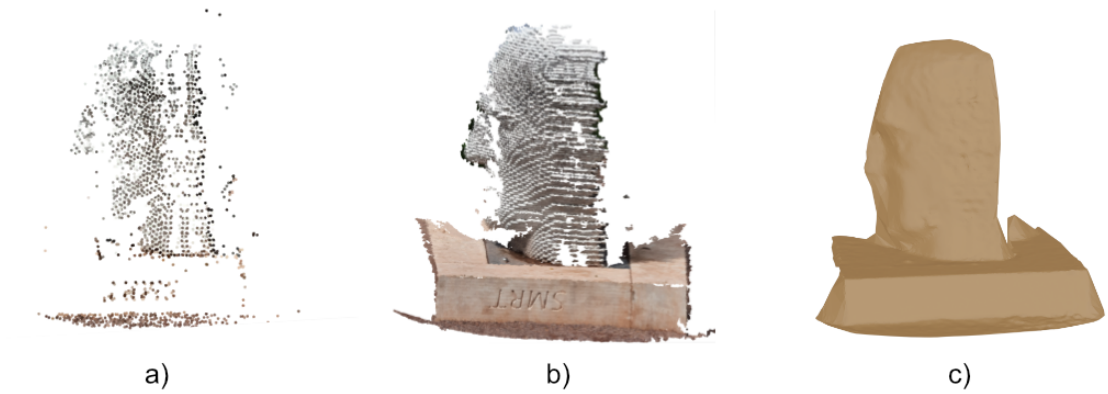
Obr. 7.8: Rekonstrukce levé strany sochy a) Řídké mračno bodů, b) Zahuštěné mračno bodů, c) Polygonální plocha

Model levé strany sochy vizuálně odpovídá objektu. Textura namapována na modelu se shoduje s barevnou strukturou na snímcích. Textura má tmavší jas vůči ostatním pohledům. Důvodem je, že tato část byla při pořizování snímků méně nasvícena přirozeným zdrojem světla a částečně na tuto část vrhá socha vlastní stín. Možným řešením tohoto problému by bylo pořídit snímky této oblasti v jinou denní dobu, případně přizpůsobit osvětlení scény vlastními zdroji světla. Problém byl částečně vyřešen zvýšením intenzity osvětlení při renderování modelu. Model odpovídá tvarům objektu.

Obr. 7.9: Levá strana modelu sochy

7.2.3 Pravá strana

Při pořizování dat pro rekonstrukci sochy z pravé strany byly pořízeny snímky ze čtyř pozic zachycujících tuto oblast. Rozlišení snímků bylo sníženo na 50 % na 1 500 x 2 000 px. Ze snímků bylo vytvořeno řídké mračno bodů obsahující 3 586 bodů v prostoru (Obr. 7.10a). Mračno bodů bylo následně zahuštěno na celkový počet 175 192 bodů (Obr. 7.10b). Polygonální rekonstrukcí povrchu byl vytvořen model zaznamenaný na Obr. 7.10c. Na tento model byla následně namapována textura ze snímků (Obr. 7.10).



Obr. 7.10: Rekonstrukce pravé strany sochy a) Řídké mračno bodů, b) Zahuštěné mračno bodů, c) Polygonální plocha

Výsledný model pravé strany sochy odpovídá tvarům objektu. Scéna byla při pořizování dat této oblasti více nasvícená vůči ostatním zpracovávaným stranám. Výsledná textura je tak světlejší než v případě předchozích pohledů. V tomto případě se také podařilo zrekonstruovat detailní oblasti sochy v podobě nápisu a odlomených částí na podstavě. Jedinou nepřesností modelu je, že při polygonální rekonstrukci povrchu byly chybně spojeny některé body na krajních částech modelu. Řešením tohoto problému je zpracování více snímků z pohledů, které by zachycovaly tyto krajní oblasti.

Obr. 7.11: Pravá strana modelu sochy

Závěr

Cílem diplomové práce bylo nastudovat problematiku mobilních aplikací a zpracování obrazu v souvislosti s 3D rekonstrukcí. Definovat vliv struktury měřené scény na kvalitu rekonstrukce. Popsat vhodné mobilní platformy a pro jednu z nich realizovat aplikaci pro 3D rekonstrukci. Pořídit sadu snímků pro testování algoritmu a zhodnotit kvalitu aplikace rekonstrukcí scén pomocí testovací sady.

V úvodní kapitole práce byla popsána projektivní geometrie. Byly uvedeny parametry kamer, které definují vztah mezi snímkem a scénou, kterou zachycuje. V další části kapitoly byla popsána epipolární geometrie, která definuje vztah mezi dvěma pohledy zaznamenávajícími stejnou scénu. V závěru kapitoly byl uveden princip triangulace, který je používán metodami prostorové rekonstrukce.

Následující kapitola se věnovala prostorové rekonstrukci. Byly popsány metody, pomocí kterých lze vytvořit ze snímků prostorový model. Úvodní část kapitoly se věnovala popisu struktury z pohybu, při které je ze snímků zaznamenaných z více pohledů získáno mračno bodů a parametry kamer. Poté byla popsána metoda vícepohledová stereoskopie, která pomocí dat získaných předchozí metodou zvýší počet bodů v prostoru. V závěru kapitoly byla popsána polygonální rekonstrukce povrchu. Polygonální rekonstrukce povrchu slouží k vytvoření povrchu z mračna bodů, který tvoří výsledný 3D model.

Další kapitola práce byla věnována mobilním platformám. Byl uveden význam mobilních platforem v souvislosti s 3D rekonstrukcí. Dále byly popsány vhodné platformy pro realizaci mobilní aplikace, kterými jsou Android a iOS. V kapitole byly také popsány vlastnosti těchto platforem.

Kapitola uzavírající teoretickou část popisovala proces získání dat. Nejprve byly definovány parametry měřené scény, které mají vliv na kvalitu výsledného modelu. Jedná se o barevnou strukturu a osvětlení scény. Následně byl popsán vhodný způsob pořízení snímků vedoucí k pořízení kvalitních dat. Ve zbylé části kapitoly byly uvedeny snímače, pomocí kterých lze na mobilních zařízeních získat dodatečné informace potřebné k prostorové rekonstrukci.

Úvodní kapitola praktické části se věnovala algoritmu rekonstrukce. Byl popsán způsob, jakým byl navržen a realizován algoritmus pro zpracování pořízených dat na základě metod popsaných v teoretické části. Dále byla popsána volba použitých nástrojů a modularita algoritmu pro použití na více platformách.

Následující kapitola popisovala realizaci mobilní aplikace pro platformu Android. Byla uvedena struktura aplikace složená z dílčích obrazovek a modulu, který představoval algoritmus rekonstrukce. V kapitole byly také popsány obrazovky, které představují dílčí kroky prostorové rekonstrukce. V závěru kapitoly bylo popsáno, jak byly zkompilovány a připojeny k aplikaci závislosti a jaká jsou cílená zařízení.

V poslední kapitole práce proběhlo otestování aplikace rekonstrukcí objektů. Testování proběhlo na mobilním zařízení střední třídy, které svým výkonem reflektuje současný trh. Pro zhodnocení kvality navržených algoritmů byla pořízena testovací sada dvou objektů. Prvním testovaným objektem byl buben. Na tomto objektu byla otestována aplikace ve vnitřních podmínkách, ve kterých bylo možné přizpůsobit měřenou scénu. Pro otestování tohoto objektu byly pořízeny snímky ze čtyř pohledů. Z těchto snímků byl vytvořen 3D model, který svým tvarem odpovídá snímanému objektu. V další části kapitoly byla aplikace otestována rekonstrukcí ve venkovních podmínkách, při kterých jsou možnosti pro přizpůsobení scény omezené. Pro otestování byla zvolena socha, která se od předchozího objektu liší rozměry a barevnou strukturou. Při rekonstrukci sochy byly vytvořeny tři modely představující rozdílné pohledy na objekt. Modely odpovídají geometrii snímaného objektu, nicméně vzhledem k rozdílnému nasvícení částí objektu, jsou některé části textury modelu světlejší než jiné. Nekonzistentní nasvícení by také bylo problematické při vytváření celkového modelu.

Využití vytvořené aplikace je pro vytvoření 3D modelu pomocí mobilního zařízení. Není nutné realizovat měřicí systém složený z více komponent. Aplikace potřebuje pro vlastní chod pouze snímky zaznamenané fotoaparátem zařízení a dostatečně výkonný hardware, kterým jsou zařízení v dnešní době vybaveny. Výsledné modely lze exportovat v 3D formátu podporovaným většinou softwarů pro práci s 3D modely. Je tak umožněno vytvořené modely dále zpracovat, nebo je přímo použít.

Budoucí práci na projektu lze rozdělit do tří oblastí. První z nich je rozšíření podporovaných platforem o platformu iOS. Další oblastí je úprava algoritmu rekonstrukce. Algoritmus detekce bodů by mohl být vylepšen invariantností vůči změně osvětlení scény. V práci byl také zjištěn problém s omezenou dostupnou operační pamětí pro aplikaci u mobilních zařízení. Optimalizace výpočtu hloubkových map by mohla vést ke zvýšení počtu snímků, které algoritmus může zpracovat. V rámci algoritmu rekonstrukce by mohla být implementována segmentace snímků, aby výsledný model neobsahoval okolní body. Poslední oblastí, kterou by se mohla zabývat budoucí práce je samotná mobilní aplikace. Aplikace by mohla podporovat exportování modelů ve více formátech. Zobrazování modelu by mohlo obsahovat základní funkce pro práci s modely, jako například ořezávání částí, nebo změnu nasvícení modelu.

Literatura

[1] KRÁTKÝ, Martin. *Vytvoření 3D modelu za pomoci laserového vzoru*. Brno, 2019. Dostupné také z: <https://www.vutbr.cz/studenti/zav-prace/detail/119313>. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Miloslav Richter.

[2] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 4th edition. United States of America: Cengage Learning, 2015. International Edition. ISBN 11-335-9369-0.

[3] AGHAJAN, Hamid K. a Andrea CAVALLARO. *Multi-camera networks: principles and applications*. Burlington: Academic Press, c2009. ISBN 978-0123746337.

[4] CYGANEK, Bogusław a J. Paul SIEBERT. *An introduction to 3D computer vision techniques and algorithms*. Chichester, U.K.: J. Wiley, 2009. ISBN 978-047-0017-043.

[5] DIEBEL, James. *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*. 2006. Dostupné z: doi:10.1.1.110.5134

[6] HARTLEY, Richard a Andrew ZISSERMAN. *Multiple view geometry in computer vision*. 2nd ed. New York: Cambridge University Press, 2003. ISBN 978-052-1540-513.

[7] TRIGGS, Bill, Philip F. MCLAUCHLAN, Richard I. HARTLEY a Andrew W. FITZGIBBON. Bundle Adjustment — A Modern Synthesis. *Vision Algorithms: Theory and Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, 12. 4. 2000, , 298-372. Lecture Notes in Computer Science. ISBN 978-3-540-67973-8. Dostupné z: doi:10.1007/3-540-44480-7_21.

[8] LI, Jianguo, Eric LI, Yurong CHEN, Lin XU a Yimin ZHANG. Bundled depth-map merging for multi-view stereo. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, 2010, 2769-2776. ISBN 978-1-4244-6984-0. Dostupné z: doi:10.1109/CVPR.2010.5540004

[9] Stereo Disparity using Semi-Global Block Matching. *MathWorks* [online]. Natick, Massachusetts, United States: The MathWorks [cit. 2021-4-29]. Dostupné z: <https://www.mathworks.com/help/visionhdl/ug/stereoscopic-disparity.html>.

[10] HIRSCHMULLER, Heiko. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2008, 30(2), 328-341. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2007.1166

[11] BLEYER, Michael, Christoph RHEMANN a Carsten ROTHER. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. *Procedings of the British Machine Vision Conference 2011*. British Machine Vision Association, 2011, 2011, 14.1-14.11. ISBN 1-901725-43-X. Dostupné z: doi:10.5244/C.25.14

[12] BETHMANN, F. a T. LUHMANN. SEMI-GLOBAL MATCHING IN OBJECT SPACE. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2015, XL-3/W2, 23-30. ISSN 2194-9034. Dostupné z: doi:10.5194/isprsarchives-XL-3-W2-23-2015

[13] SHEN, Shuhan. Accurate Multiple View 3D Reconstruction Using Patch-Based Stereo for Large-Scale Scenes. *IEEE Transactions on Image Processing*. 2013, 22(5), 1901-1914. ISSN 1057-7149. Dostupné z: doi:10.1109/TIP.2013.2237921

[14] KAZHDAN, Michael, Matthew BOLITHO a Hoppe HUGUES. Poisson Surface Reconstruction. *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Cagliari, Sardinia, Italy: Eurographics Association, 2006, SGP '06, 61–70. Dostupné z: doi:10.5555/1281957.1281965

[15] BOLITHO, Matthew, Michael KAZHDAN, Randal BURNS a Hugues HOPPE. Parallel Poisson Surface Reconstruction. *Advances in Visual Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, 2009, 678-689. Lecture Notes in Computer Science. ISBN 978-3-642-10330-8. Dostupné z: doi:10.1007/978-3-642-10331-5_63

[16] BERG, Mark de, Otfried CHEONG, Marc van KREVELD a Mark OVERMARS. *Computational Geometry: Algorithms and Applications*. Third Edition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77974-2.

[17] SINHA, Sudipta N., Philippos MORDOHAI a Marc POLLEFEYS. Multi-View Stereo via Graph Cuts on the Dual of an Adaptive Tetrahedral Mesh. *IEEE 11th International Conference on Computer Vision*. 2007, 1-8. ISBN 978-1-4244-1630-1. Dostupné z: doi:10.1109/ICCV.2007.4408997

[18] NISTER, David. An efficient solution to the five-point relative pose problem. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*. IEEE Comput. Soc, 2003, II-195-202. ISBN 0-7695-1900-8. Dostupné z: doi:10.1109/CVPR.2003.1211470

[19] GONZALEZ, Rafael C. a Richard E. WOODS. *Digital Image Processing*. 4th edition. London: Pearson, 2018. ISBN 978-1292223049.

[20] LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*. 2004, 60(2), 91-110. ISSN 0920-5691. Dostupné z: doi:10.1023/B:VISI.0000029664.99615.94

[21] BAY, Herbert, Andreas ESS, Tinne TUYTELAARS a Luc VAN GOOL. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*. 2008, 110(3), 346-359. ISSN 10773142. Dostupné z: doi:10.1016/j.cviu.2007.09.014

[22] ALCANTARILLA, Pablo Fernández, Adrien BARTOLI a Andrew J. DAVISON. KAZE Features. *Computer Vision – ECCV 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 214-227. Lecture Notes in Computer Science. ISBN 978-3-642-33782-6. Dostupné z: doi:10.1007/978-3-642-33783-3_16

[23] ALCANTARILLA, Pablo, Jesus NUEVO a Adrien BARTOLI. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. *Procedings of the British Machine Vision Conference 2013*. British Machine Vision Association, 2013, 13.1-13.11. ISBN 1-901725-49-9. Dostupné z: doi:10.5244/C.27.13

[24] HIDALGO, Franco a Thomas BRÄUNL. Evaluation of Several Feature Detectors/Extractors on Underwater Images towards vSLAM. *Sensors*. 2020, 20(15). ISSN 1424-8220. Dostupné z: doi:10.3390/s20154343

[25] SNAVELY, Noah, Steven M. SEITZ a Richard SZELISKI. Modeling the World from Internet Photo Collections. *International Journal of Computer Vision*. 2008, 80(2), 189-210. ISSN 0920-5691. Dostupné z: doi:10.1007/s11263-007-0107-3

[26] LI, Jianguo, Eric LI, Yurong CHEN, Lin XU a Yimin ZHANG. Bundled depth-map merging for multi-view stereo. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, 2769-2776. ISBN 978-1-4244-6984-0. Dostupné z: doi:10.1109/CVPR.2010.5540004

[27] Mobile Operating System Market Share Worldwide. *StatCounter* [online]. Dublin: StatCounter, 2021 [cit. 2021-05-01]. Dostupné z: <`https://gs.statcounter.com/os-market-share/mobile/worldwide`>.

[28] GRIFFITH, Chris. *Mobile App Development with Ionic, Revised Edition: Cross-Platform Apps with Ionic, Angular, and Cordova*. Newton: O'Reilly Media, 2017. ISBN 1491998121.

[29] Platform Architecture. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/guide/platform>.

[30] Android Studio. *Android Developers* [software]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/studio>.

[31] SDK Tools release notes. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/studio/releases/sdk-tools>.

[32] Android NDK. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/ndk>.

[33] Google Play. *Google* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://play.google.com>.

[34] Galaxy Store. *Samsung* [online]. Soul: Samsung, 2021 [cit. 2021-05-01]. Dostupné z:
<https://www.samsung.com/cz/apps/galaxy-store>.

[35] Amazon Appstore. *Amazon* [online]. Washington: Amazon.com, 2021 [cit. 2021-05-01]. Dostupné z:
<https://www.amazon.com/gp/mas/get/amazonapp>.

[36] Application Fundamentals. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/guide/components/fundamentals>.

[37] *Gradle* [software]. San Francisco: Gradle, 2021 [cit. 2021-05-01]. Dostupné z:
<https://gradle.org/>.

[38] Configure your build. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/studio/build>.

[39] Sign your app. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z:
<https://developer.android.com/studio/publish/app-signing>.

[40] Android App Bundle. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/platform/technology/app-bundle>.

[41] App Manifest Overview. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/guide/topics/manifest/manifest-intro>.

[42] App resources overview. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/guide/topics/resources/providing-resources>.

[43] *CMake* [online]. New York: Kitware, 2021 [cit. 2021-05-01]. Dostupné z: <https://cmake.org/>.

[44] Xcode. *Apple Developer* [software]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/xcode>.

[45] App Store. *Apple* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://www.apple.com/cz/ios/app-store>.

[46] TAMMA, Rohit, Oleg SKULKIN, Heather MAHALIK a Satish BOMMISETTY. *Practical Mobile Forensics: Forensically investigate and analyze iOS, Android, and Windows 10 devices*. 4th Edition. Birmingham: Packt, 2020. ISBN 978-1838647520.

[47] What are Frameworks? *Documentation Archive* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html>.

[48] What Is Cocoa? *Documentation Archive* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html>.

[49] LEE, Wei-Meng. *Beginning iOS 5 Application Development*. Birmingham: Wrox, 2012. ISBN 978-1118144251.

[50] Model-View-Controller. *Documentation Archive* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html>.

[51] Information Property List. *Apple Developer Documentation* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/documentation/bundleresources/information_property_list>.

[52] BAECHLER, Oscar a Xury GREER. *Blender 3D By Example: A project-based guide to learning the latest Blender 3D, EEVEE rendering engine, and Grease Pencil*. 2nd Edition. Birmingham: Packt, 2020. ISBN 9781789612561.

[53] RAY, Sidney. *Scientific Photography and Applied Imaging*. London: Routledge, 1999. ISBN 9780240513232.

[54] SensorManager. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/reference/android/hardware/SensorManager>.

[55] CMMotionManager. *Apple Developer* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/documentation/coremotion/cmmotionmanager>.

[56] Position sensors. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/guide/topics/sensors/sensors_position>.

[57] BASAR, Tamer. *A New Approach to Linear Filtering and Prediction Problems* [online]. Control Theory. IEEE, 2000, 2009. ISBN 9780470544334. Dostupné z: doi:10.1109/9780470544334.ch9.

[58] Getting Raw Accelerometer Events. *Apple Developer* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events>.

[59] Getting Raw Gyroscope Events. *Apple Developer* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z:

<https://developer.apple.com/documentation/coremotion/getting_raw_gyroscope_events>.

[60] Motion sensors. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/guide/topics/sensors/sensors_motion>.

[61] ARCore overviews. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developers.google.com/ar/discover>.

[62] Augmented Reality. *Apple Developer* [online]. Cupertino: Apple, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.apple.com/augmented-reality/>.

[63] FISCHLER, Martin A. a Robert C. BOLLES. Random sample consensus. *Communications of the ACM*. 1981, 24(6), 381-395. ISSN 0001-0782. Dostupné z: doi:10.1145/358669.358692

[64] ARCore supported devices. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developers.google.com/ar/discover/supported-devices>.

[65] CameraCharacteristics. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/reference/android/hardware/camera2/CameraCharacteristics>.

[66] *OpenCV* [online]. Santa Clara: Intel, 2021 [cit. 2021-05-01]. Dostupné z: <https://opencv.org>.

[67] Structure From Motion. *OpenCV* [online]. Santa Clara: Intel, 2021 [cit. 2021-05-01]. Dostupné z: <https://docs.opencv.org/master/d8/d8c/group__sfm.html>.

[68] Libmv *Blender Developer* [online]. Amsterdam: Blender Foundation, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.blender.org/tag/libmv/>.

[69] *OpenMVS* [online]. Bucharest: SEACAVE SRL, 2021 [cit. 2021-05-01]. Dostupné z: <http://cdcseacave.github.io/openMVS>.

[70] Build ARCore apps without OpenGL. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developers.google.com/sceneform>.

[71] Create dynamic lists with RecyclerView. *Android Developers* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://developer.android.com/guide/topics/ui/layout/recyclerview>.

[72] BENOÎT, Jacob a Gaël GUENNEBAUD. *Eigen* [online]. 2021 [cit. 2021-5-1]. Dostupné z: <https://eigen.tuxfamily.org/>.

[73] *Google Logging Library* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://github.com/google/glog>.

[74] *gflags* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <https://github.com/gflags/gflags>.

[75] *Ceres Solver* [online]. Mountain View: Google, 2021 [cit. 2021-05-01]. Dostupné z: <http://ceres-solver.org/>.

[76] *The Computational Geometry Algorithms Library* [online]. The CGAL Project, 2021 [cit. 2021-05-01]. Dostupné z: <https://www.cgal.org/>.

[77] *Boost C++ Libraries* [online]. Boost, 2021 [cit. 2021-05-01]. Dostupné z: <https://www.boost.org/>.

[78] *Visualization and Computer Graphics Library* [online]. 2021 [cit. 2021-05-01]. Dostupné z: <http://vcglib.net/>.

[79] Galaxy A51. *Samsung* [online]. Soul: Samsung, 2021, [cit. 2021-05-01]. Dostupné z: <https://www.samsung.com/cz/smartphones/galaxy-a/galaxy-a51-black-128gb-sm-a515fzkveue/>.

Seznam symbolů a zkratek

2D	Dvourozměrný
3D	Trojrozměrný
AAB	Android App Bundle
APK	Application Package
API	Application Programming Interface
AR	Augmented Reality
ARM	Advanced RISC Machines
CPU	Central Processing Unit
DEX	Dalvik Executable
DSL	Domain Specific Language
glb	Graphics Language Transmission Format Binary
glTF	Graphics Language Transmission Format
GPS	Global Positioning System
ID	Identifikátor
NDK	Native Development Kit
OS	Operační Systém
RISC	Reduced instruction set Computing
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features

Seznam příloh

A Obsah elektronické přílohy

- `/` — kořenový adresář přílohy
  - `Dataset` — testovací sady snímků
    - `Buben` — sada snímků bubnu
    - `Socha` — sada snímků sochy
  - `Instalační soubory` — instalační soubory aplikace
    - `reconstruction-arm64-v8a.apk` — instalační soubor aplikace pro arm64-v8a
  - `Modely` — 3D modely
    - `buben.glb` — model bubnu
    - `socha-leva.glb` — model levé části sochy
    - `socha-prava.glb` — model pravé části sochy
    - `socha-predni.glb` — model přední části sochy
  - `Video` — záznamy obrazovky z testování aplikace
    - `test-buben.mp4` — video rekonstrukce bubnu
    - `test-socha-leva.mp4` — video rekonstrukce levé části sochy
    - `test-socha-prava.mp4` — video rekonstrukce pravé části sochy
    - `test-socha-predni.mp4` — video rekonstrukce přední části sochy
  - `Zdrojové soubory` — použité zdrojové soubory
    - `Aplikace` — zdrojové soubory mobilní aplikace
    - `Knihovny` — zkompilované knihovny
    - `Skripty` — skripty pro překlad knihoven
  - `Mobilní aplikace pro 3D rekonstrukci.pdf` — text práce